Factoring integers, Producing primes and the RSA cryptosystem

Harish-Chandra Research Institute

Allahabad (UP), INDIA

February, 2005

$$RSA_{2048} = 25195908475657893494027183240048398571429282126204$$

032027777137836043662020707595556264018525880784406918290641249
515082189298559149176184502808489120072844992687392807287776735
971418347270261896375014971824691165077613379859095700097330459
748808428401797429100642458691817195118746121515172654632282216
869987549182422433637259085141865462043576798423387184774447920
739934236584823824281198163815010674810451660377306056201619676
256133844143603833904414952634432190114657544454178424020924616
515723350778707749817125772467962926386356373289912154831438167
899885040445364023527381951378636564391212010397122822120720357

RSA_{2048} = 25195908475657893494027183240048398571429282126204
032027777137836043662020707595556264018525880784406918290641249
515082189298559149176184502808489120072844992687392807287776735
971418347270261896375014971824691165077613379859095700097330459
748808428401797429100642458691817195118746121515172654632282216
869987549182422433637259085141865462043576798423387184774447920
739934236584823824281198163815010674810451660377306056201619676
256133844143603833904414952634432190114657544454178424020924616
515723350778707749817125772467962926386356373289912154831438167
899885040445364023527381951378636564391212010397122822120720357

RSA_{2048} is a 617 (decimal) digit number

$$RSA_{2048} = 25195908475657893494027183240048398571429282126204$$
$$032027777137836043662020707595556264018525880784406918290641249$$
$$515082189298559149176184502808489120072844992687392807287776735$$
$$971418347270261896375014971824691165077613379859095700097330459$$
$$748808428401797429100642458691817195118746121515172654632282216$$
$$869987549182422433637259085141865462043576798423387184774447920$$
$$739934236584823824281198163815010674810451660377306056201619676$$
$$256133844143603833904414952634432190114657544454178424020924616$$
$$515723350778707749817125772467962926386356373289912154831438167$$
$$899885040445364023527381951378636564391212010397122822120720357$$

RSA_{2048} is a 617 (decimal) digit number

http://www.rsa.com/rsalabs/challenges/factoring/numbers.html/

$$RSA_{2048} = p \cdot q, \quad p, q \approx 10^{308}$$

$$RSA_{2048} = p \cdot q, \qquad p, q \approx 10^{308}$$

PROBLEM: *Compute p and q*

$$RSA_{2048} = p \cdot q, \quad p, q \approx 10^{308}$$

PROBLEM: *Compute* p *and* q

PRICE: 200.000 US\$ ($\sim 87,36,000$ Indian Rupee)!!

$$RSA_{2048}{=}p\cdot q,\qquad p,q\approx 10^{308}$$

PROBLEM: *Compute p and q*

PRICE: 200.000 US\$ ($\sim 87,36,000$ Indian Rupee)!!

Theorem. If $a\in\mathbb{N}\quad\exists!\ p_1<p_2<\cdots<p_k$ *primes* s.t. $a=p_1^{\alpha_1}\cdots p_k^{\alpha_k}$

$$RSA_{2048}{=}p\cdot q,\quad p,q\approx 10^{308}$$

PROBLEM: *Compute p and q*

PRICE: 200.000 US$ ($\sim 87,36,000$ Indian Rupee)!!

Theorem. If $a\in\mathbb{N}\quad \exists!\ p_1<p_2<\cdots<p_k$ *primes* s.t. $a=p_1^{\alpha_1}\cdots p_k^{\alpha_k}$

Regrettably: RSAlabs believes that factoring in one year requires:

number	computers	memory
RSA_{1620}	1.6×10^{15}	120 Tb
RSA_{1024}	$342,000,000$	170 Gb
RSA_{760}	215,000	4Gb.

http://www.rsa.com/rsalabs/challenges/factoring/numbers.html

http://www.rsa.com/rsalabs/challenges/factoring/numbers.html

Challenge Number	Prize ($US)
RSA_{576}	$10,000
RSA_{640}	$20,000
RSA_{704}	$30,000
RSA_{768}	$50,000
RSA_{896}	$75,000
RSA_{1024}	$100,000
RSA_{1536}	$150,000
RSA_{2048}	$200,000

http://www.rsa.com/rsalabs/challenges/factoring/numbers.html

Challenge Number	Prize ($US)	Status
RSA_{576}	$10,000	Factored December 2003
RSA_{640}	$20,000	Not Factored
RSA_{704}	$30,000	Not Factored
RSA_{768}	$50,000	Not Factored
RSA_{896}	$75,000	Not Factored
RSA_{1024}	$100,000	Not Factored
RSA_{1536}	$150,000	Not Factored
RSA_{2048}	$200,000	Not Factored

History of the "Art of Factoring"

History of the "Art of Factoring"

- 220 BC Greeks (Eratosthenes of Cyrene)

History of the "Art of Factoring"

- 220 BC Greeks (Eratosthenes of Cyrene)
- 1730 Euler $2^{2^5} + 1 = 641 \cdot 6700417$

History of the "Art of Factoring"

- 220 BC Greeks (Eratosthenes of Cyrene)
- 1730 Euler $2^{2^5} + 1 = 641 \cdot 6700417$
- 1750–1800 Fermat, Gauss (Sieves - Tables)

History of the "Art of Factoring"

- 220 BC Greeks (Eratosthenes of Cyrene)
- 1730 Euler $2^{2^5}+1=641\cdot 6700417$
- 1750–1800 Fermat, Gauss (Sieves - Tables)
- 1880 Landry & Le Lasseur:

$$2^{2^6}+1=274177\times 67280421310721$$

History of the "Art of Factoring"

- 220 BC Greeks (Eratosthenes of Cyrene)
- 1730 Euler $2^{2^5}+1=641\cdot 6700417$
- 1750–1800 Fermat, Gauss (Sieves - Tables)
- 1880 Landry & Le Lasseur:

$$2^{2^6}+1=274177\times 67280421310721$$

- 1919 Pierre and Eugène Carissan (Factoring Machine)

History of the "Art of Factoring"

- 220 BC Greeks (Eratosthenes of Cyrene)
- 1730 Euler $2^{2^5}+1 = 641 \cdot 6700417$
- 1750–1800 Fermat, Gauss (Sieves - Tables)
- 1880 Landry & Le Lasseur:

$$2^{2^6}+1 = 274177 \times 67280421310721$$

- 1919 Pierre and Eugène Carissan (Factoring Machine)
- 1970 Morrison & Brillhart

$$2^{2^7}+1 = 59649589127497217 \times 5704689200685129054721$$

History of the "Art of Factoring"

- 220 BC Greeks (Eratosthenes of Cyrene)
- 1730 Euler $2^{2^5}+1 = 641 \cdot 6700417$
- 1750–1800 Fermat, Gauss (Sieves - Tables)
- 1880 Landry & Le Lasseur:

$$2^{2^6}+1 = 274177 \times 67280421310721$$

- 1919 Pierre and Eugène Carissan (Factoring Machine)
- 1970 Morrison & Brillhart

$$2^{2^7}+1 = 59649589127497217 \times 5704689200685129054721$$

- 1982 Quadratic Sieve **QS** (Pomerance) $\rightsquigarrow$ Number Fields Sieve **NFS**

History of the "Art of Factoring"

- 220 BC Greeks (Eratosthenes of Cyrene)
- 1730 Euler $2^{2^5}+1=641\cdot 6700417$
- 1750–1800 Fermat, Gauss (Sieves - Tables)
- 1880 Landry & Le Lasseur:

$$2^{2^6}+1=274177\times 67280421310721$$

- 1919 Pierre and Eugène Carissan (Factoring Machine)
- 1970 Morrison & Brillhart

$$2^{2^7}+1=59649589127497217\times 5704689200685129054721$$

- 1982 Quadratic Sieve **QS** (Pomerance) $\rightsquigarrow$ Number Fields Sieve **NFS**
- 1987 Elliptic curves factoring **ECF** (Lenstra)

Carissan's ancient Factoring Machine

Carissan's ancient Factoring Machine

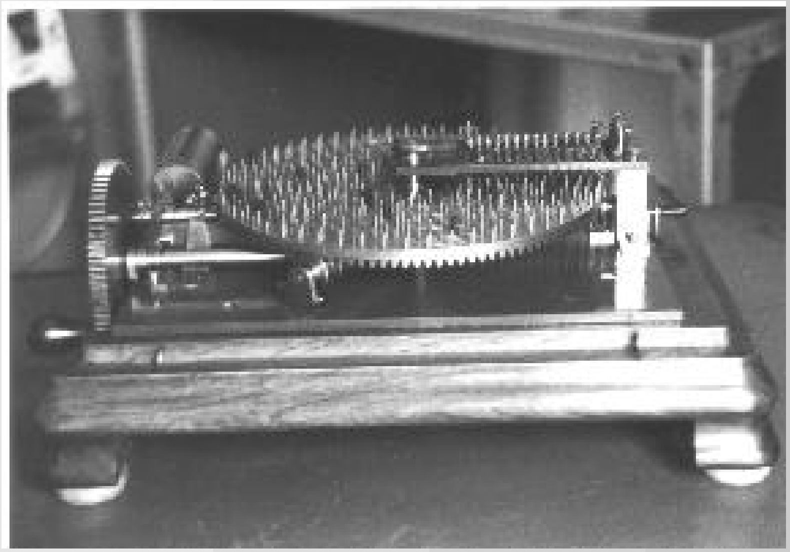


Figure 1: Conservatoire Nationale des Arts et Métiers in Paris

Carissan's ancient Factoring Machine

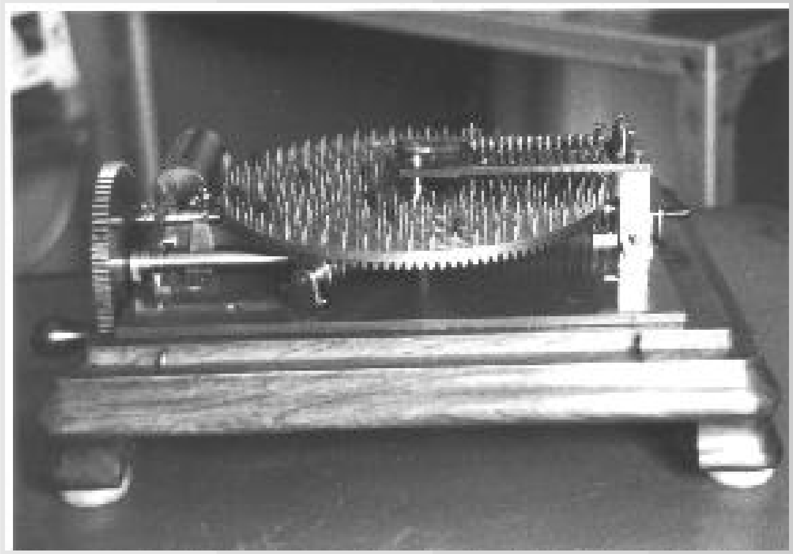


Figure 1: Conservatoire Nationale des Arts et Métiers in Paris

`http://www.math.uwaterloo.ca/ shallit/Papers/carissan.html`

Figure 2: Lieutenant Eugène Carissan

Figure 2: Lieutenant Eugène Carissan

$225058681 = 229 \times 982789$	2 minutes
$3450315521 = 1409 \times 2418769$	3 minutes
$3570537526921 = 841249 \times 4244329$	18 minutes

Contemporary Factoring

Contemporary Factoring

❶ 1994, Quadratic Sieve (QS): (8 months, 600 voluntaries, 20 countries)
D.Atkins, M. Graff, A. Lenstra, P. Leyland

$$
\begin{aligned}
RSA_{129} = {} & 114381625757888867669235779976146612010218296721242362562561842935706 \\
& 935245733897830597123563958705058989075147599290026879543541 = \\
= {} & 3490529510847650949147849619903898133417764638493387843990820577 \times \\
& 32769132993266709549961988190834461413177642967992942539798288533
\end{aligned}
$$

Contemporary Factoring

1. 1994, Quadratic Sieve (QS): (8 months, 600 voluntaries, 20 countries)
   D.Atkins, M. Graff, A. Lenstra, P. Leyland

$$
\begin{aligned}
RSA_{129} &= 114381625757888867669235779976146612010218296721242362562561842935706\\
&\quad 935245733897830597123563958705058989075147599290026879543541 =\\
&= 3490529510847650949147849619903898133417764638493387843990820577 \times\\
&\quad 32769132993266709549961988190834461413177642967992942539798288533
\end{aligned}
$$

2. (February 2 1999), Number Fields Sieve (NFS): (160 Sun, 4 months)

$$
\begin{aligned}
RSA_{155} &= 109417386415705274218097073220403576120037329454492059909138421314763499842\\
&\quad 88934784717997257891267332497625752899781833797076537244027146743531593354333897 =\\
&= 102639592829741105772054196573991675900716567808038066803341933521790711307779 \times\\
&\quad 106603488380168454820927220360012878679207958575989291522270608237193062808643
\end{aligned}
$$

Contemporary Factoring

1. 1994, Quadratic Sieve (QS): (8 months, 600 voluntaries, 20 countries) D.Atkins, M. Graff, A. Lenstra, P. Leyland

$$RSA_{129} = 114381625757888867669235779976146612010218296721242362562561842935706 935245733897830597123563958705058989075147599290026879543541 =$$
$$= 3490529510847650949147849619903898133417764638493387843990820577 \times 32769132993266709549961988190834461413177642967992942539798288533$$

2. (February 2 1999), Number Fields Sieve (NFS): (160 Sun, 4 months)

$$RSA_{155} = 109417386415705274218097073220403576120037329454492059909138421314763499842 88934784717997257891267332497625752899781833797076537244027146743531593354333897 =$$
$$= 102639592829741105772054196573991675900716567808038066803341933521790711307779 \times 106603488380168454820927220360012878679207958575989291522270608237193062808643$$

3. (December 3, 2003) (NFS): J. Franke et al. (174 decimal digits)

$$RSA_{576} = 188198812920607963838697239461650439807163563379417382700763356422988859715234 665485319060606504743045317388011303396716199692321205734031879550656996221305168759307650257059 =$$
$$= 398075086424064937397125500550386491199064362342526708406385189575946388957261768583317 \times 472772146107435302536223071973048224632914695302097116459852171130520711256363590397527$$

Contemporary Factoring

1. 1994, Quadratic Sieve (QS): (8 months, 600 voluntaries, 20 countries)
   D.Atkins, M. Graff, A. Lenstra, P. Leyland

$$RSA_{129} = 114381625757888867669235779976146612010218296721242362562561842935706$$
$$935245733897830597123563958705058989075147599290026879543541 =$$
$$= 3490529510847650949147849619903898133417764638493387843990820577 \times$$
$$32769132993266709549961988190834461413177642967992942539798288533$$

2. (February 2 1999), Number Fields Sieve (NFS): (160 Sun, 4 months)

$$RSA_{155} = 109417386415705274218097073220403576120037329454492059909138421314763499842$$
$$88934784717997257891267332497625752899781833797076537244027146743531593354333897 =$$
$$= 102639592829741105772054196573991675900716567808038066803341933521790711307779 \times$$
$$106603488380168454820927220360012878679207958575989291522270608237193062808643$$

3. (December 3, 2003) (NFS): J. Franke et al. (174 decimal digits)

$$RSA_{576} = 1881988129206079638386972394616504398071635633794173827007633564229888597152346$$
$$65485319060606504743045317388011303396716199692321205734031879550656996221305168759307650257059 =$$
$$= 398075086424064937397125500550386491199064362342526708406385189575946388957261768583317 \times$$
$$472772146107435302536223071973048224632914695302097116459852171130520711256363590397527$$

4. Elliptic curves factoring: introduced by da H. Lenstra. suitable to find prime factors with 50 digits (small)

Contemporary Factoring

1. 1994, Quadratic Sieve (QS): (8 months, 600 voluntaries, 20 countries) D.Atkins, M. Graff, A. Lenstra, P. Leyland

$$RSA_{129} = 114381625757888867669235779976146612010218296721242362562561842935706$$
$$935245733897830597123563958705058989075147599290026879543541 =$$
$$= 3490529510847650949147849619903898133417764638493387843990820577 \times$$
$$32769132993266709549961988190834461413177642967992942539798288533$$

2. (February 2 1999), Number Fields Sieve (NFS): (160 Sun, 4 months)

$$RSA_{155} = 109417386415705274218097073220403576120037329454492059909138421314763499842$$
$$88934784717997257891267332497625752899781833797076537244027146743531593354333897 =$$
$$= 102639592829741105772054196573991675900716567808038066803341933521790711307779 \times$$
$$106603488380168454820927220360012878679207958575989291522270608237193062808643$$

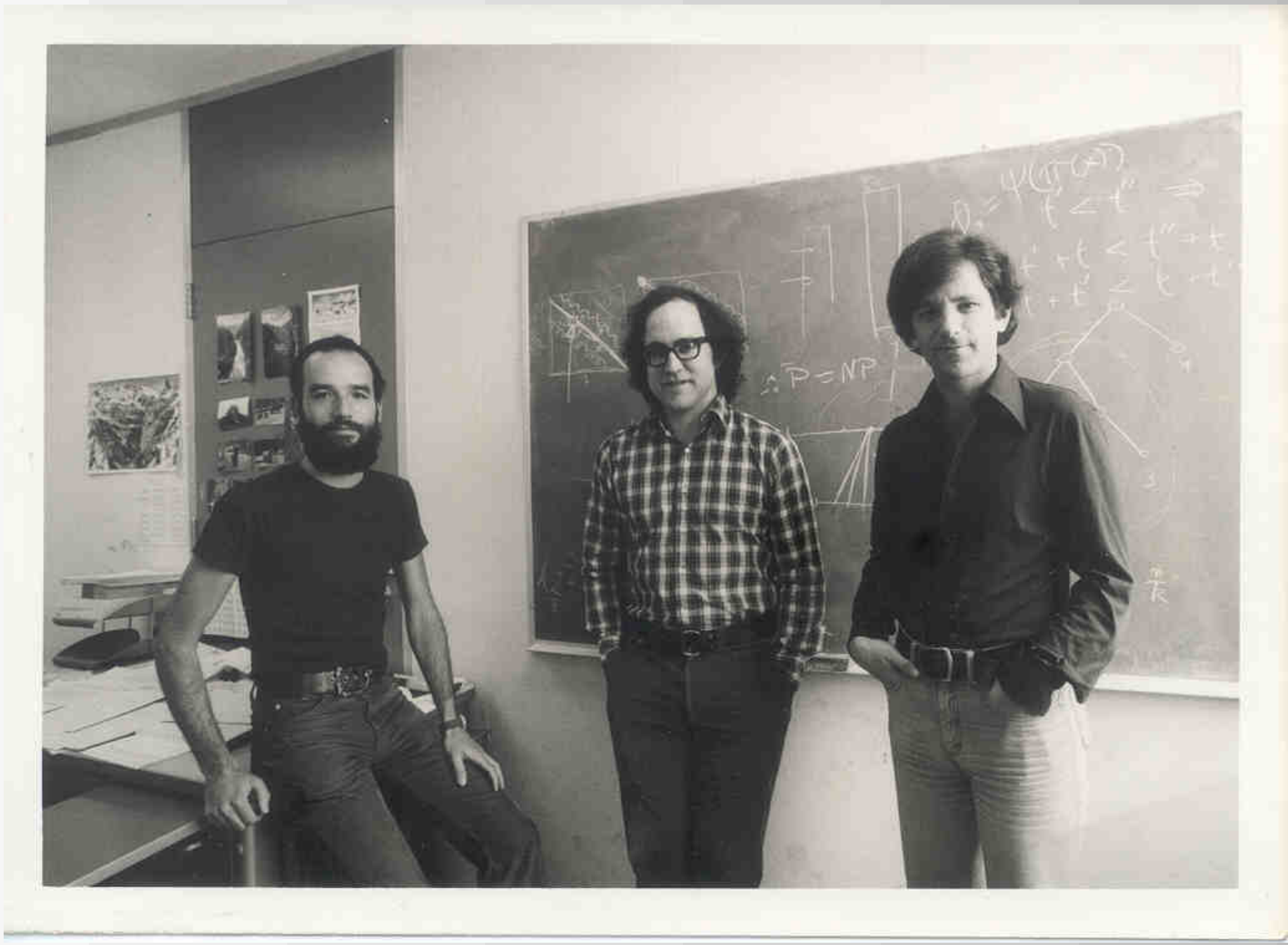
3. (December 3, 2003) (NFS): J. Franke et al. (174 decimal digits)

$$RSA_{576} = 188198812920607963838697239461650439807163563379417382700763356422988859715234$$
$$6548531906060650474304531738801130339671619969232120573403187955065699622130516875930765025705 9 =$$
$$= 398075086424064937397125500550386491199064362342526708406385189575946388957261768583317 \times$$
$$472772146107435302536223071973048224632914695302097116459852171130520711256363590397527$$

4. Elliptic curves factoring: introduced by da H. Lenstra. suitable to find prime factors with 50 digits (small)

All: "sub–exponential running time"

RSA



Adi Shamir, Ron L. Rivest, Leonard Adleman (1978)

The RSA cryptosystem

The RSA cryptosystem

1978 R. L. Rivest, A. Shamir, L. Adleman (Patent expired in 1998)

The RSA cryptosystem

1978 R. L. Rivest, A. Shamir, L. Adleman (Patent expired in 1998)

Problem: Alice wants to send the message $\mathcal{P}$ to Bob so that Charles cannot read it

The RSA cryptosystem

1978 R. L. Rivest, A. Shamir, L. Adleman (Patent expired in 1998)

Problem: Alice wants to send the message $\mathcal{P}$ to Bob so that Charles cannot read it

$$\boxed{\begin{array}{ccc} A\ (Alice) & \longrightarrow & B\ (Bob) \\ & \uparrow & \\ & C\ (Charles) & \end{array}}$$

The RSA cryptosystem

1978 R. L. Rivest, A. Shamir, L. Adleman (Patent expired in 1998)

Problem: Alice wants to send the message $\mathcal{P}$ to Bob so that Charles cannot read it

$$\begin{array}{ccc} A\ (Alice) & \longrightarrow & B\ (Bob) \\ & \uparrow & \\ & C\ (Charles) & \end{array}$$

❶

❷

❸

❹

The RSA cryptosystem

1978 R. L. Rivest, A. Shamir, L. Adleman (Patent expired in 1998)

Problem: Alice wants to send the message $\mathcal{P}$ to Bob so that Charles cannot read it

$$\begin{array}{ccc} A\ (Alice) & \longrightarrow & B\ (Bob) \\ & \uparrow & \\ & C\ (Charles) & \end{array}$$

1. Key generation — Bob has to do it
2.
3.
4.

The RSA cryptosystem

1978 R. L. Rivest, A. Shamir, L. Adleman (Patent expired in 1998)

Problem: Alice wants to send the message $\mathcal{P}$ to Bob so that Charles cannot read it

$$\boxed{\begin{array}{ccc} A\ (Alice) & \longrightarrow & B\ (Bob) \\ & \uparrow & \\ & C\ (Charles) & \end{array}}$$

1. Key generation — Bob has to do it
2. Encryption — Alice has to do it
3.
4.

The RSA cryptosystem

1978 R. L. Rivest, A. Shamir, L. Adleman (Patent expired in 1998)

Problem: Alice wants to send the message $\mathcal{P}$ to Bob so that Charles cannot read it

$$\begin{array}{ccc} A\ (Alice) & \longrightarrow & B\ (Bob) \\ & \uparrow & \\ & C\ (Charles) & \end{array}$$

1. Key generation — Bob has to do it
2. Encryption — Alice has to do it
3. Decryption — Bob has to do it
4.

The RSA cryptosystem

1978 R. L. Rivest, A. Shamir, L. Adleman (Patent expired in 1998)

Problem: Alice wants to send the message $\mathcal{P}$ to Bob so that Charles cannot read it

$$\begin{array}{ccc} A\ (Alice) & \longrightarrow & B\ (Bob) \\ & \uparrow & \\ & C\ (Charles) & \end{array}$$

1. KEY GENERATION — Bob has to do it
2. ENCRYPTION — Alice has to do it
3. DECRYPTION — Bob has to do it
4. ATTACK — Charles would like to do it

Bob: Key generation

Bob: Key generation

✍

✍

✍

✍

✍

Bob: Key generation

- He chooses randomly p and q primes $(p, q \approx 10^{100})$
-
-
-
-

Bob: Key generation

- He chooses randomly p and q primes $\quad (p, q \approx 10^{100})$
- He computes $\quad M = p \times q,\ \varphi(M) = (p-1) \times (q-1)$
-
-
-

Bob: Key generation

- He chooses randomly p and q primes $(p, q \approx 10^{100})$
- He computes $M = p \times q$, $\varphi(M) = (p-1) \times (q-1)$
- He chooses an integer e s.t.

Bob: Key generation

- He chooses randomly p and q primes $(p, q \approx 10^{100})$
- He computes $M = p \times q$, $\varphi(M) = (p-1) \times (q-1)$
- He chooses an integer e s.t.

$$0 \leq e \leq \varphi(M) \quad \text{and} \quad \gcd(e, \varphi(M)) = 1$$

-
-

Bob: Key generation

- He chooses randomly p and q primes $(p, q \approx 10^{100})$
- He computes $M = p \times q$, $\varphi(M) = (p-1) \times (q-1)$
- He chooses an integer e s.t.

$$0 \le e \le \varphi(M) \quad \text{and} \quad \gcd(e, \varphi(M)) = 1$$

NOTE. One could take $e = 3$ and $p \equiv q \equiv 2 \bmod 3$

-
-

Bob: Key generation

- He chooses randomly p and q primes $(p, q \approx 10^{100})$
- He computes $M = p \times q$, $\varphi(M) = (p-1) \times (q-1)$
- He chooses an integer e s.t.

$$0 \leq e \leq \varphi(M) \text{ and } \gcd(e, \varphi(M)) = 1$$

NOTE. One could take $e = 3$ and $p \equiv q \equiv 2 \bmod 3$

Experts recommend $e = 2^{16} + 1$

Bob: Key generation

- He chooses randomly p and q primes $(p, q \approx 10^{100})$
- He computes $M = p \times q$, $\varphi(M) = (p-1) \times (q-1)$
- He chooses an integer e s.t.

$$0 \le e \le \varphi(M) \quad \text{and} \quad \gcd(e, \varphi(M)) = 1$$

Note. One could take $e = 3$ and $p \equiv q \equiv 2 \bmod 3$

Experts recommend $e = 2^{16} + 1$

- He computes arithmetic inverse d of e modulo $\varphi(M)$
-

Bob: Key generation

- He chooses randomly p and q primes $(p, q \approx 10^{100})$
- He computes $M = p \times q$, $\varphi(M) = (p-1) \times (q-1)$
- He chooses an integer e s.t.

  $$0 \leq e \leq \varphi(M) \text{ and } \gcd(e, \varphi(M)) = 1$$

  NOTE. One could take $e = 3$ and $p \equiv q \equiv 2 \bmod 3$

  Experts recommend $e = 2^{16} + 1$
- He computes arithmetic inverse d of e modulo $\varphi(M)$

  (i.e. $d \in \mathbb{N}$ (unique $\leq \varphi(M)$) s.t. $e \times d \equiv 1 \pmod{\varphi(M)}$)
-

Bob: Key generation

- He chooses randomly p and q primes $(p, q \approx 10^{100})$
- He computes $M = p \times q$, $\varphi(M) = (p-1) \times (q-1)$
- He chooses an integer e s.t.

  $$0 \le e \le \varphi(M) \text{ and } \gcd(e, \varphi(M)) = 1$$

  NOTE. One could take $e = 3$ and $p \equiv q \equiv 2 \bmod 3$

  Experts recommend $e = 2^{16} + 1$
- He computes arithmetic inverse d of e modulo $\varphi(M)$

  (i.e. $d \in \mathbb{N}$ (unique $\le \varphi(M)$) s.t. $e \times d \equiv 1 \pmod{\varphi(M)}$)
- Publishes (M, e) **public key** and hides **secret key** d

Bob: Key generation

- ✍ He chooses randomly p and q primes $(p, q \approx 10^{100})$
- ✍ He computes $M = p \times q$, $\varphi(M) = (p-1) \times (q-1)$
- ✍ He chooses an integer e s.t.

  $$0 \le e \le \varphi(M) \quad \text{and} \quad \gcd(e, \varphi(M)) = 1$$

  NOTE. One could take $e = 3$ and $p \equiv q \equiv 2 \bmod 3$

  Experts recommend $e = 2^{16} + 1$
- ✍ He computes arithmetic inverse d of e modulo $\varphi(M)$

  (i.e. $d \in \mathbb{N}$ (unique $\le \varphi(M)$) s.t. $e \times d \equiv 1 \pmod{\varphi(M)}$)
- ✍ Publishes (M, e) **public key** and hides **secret key** d

Problem: How does Bob do all this?- We will go came back to it!

Alice: Encryption

Alice: Encryption

Represent the message $\mathcal{P}$ as an element of $\mathbb{Z}/M\mathbb{Z}$

Alice: Encryption

Represent the message $\mathcal{P}$ as an element of $\mathbb{Z}/M\mathbb{Z}$

(for example) $\boxed{A \leftrightarrow 1 \quad B \leftrightarrow 2 \quad C \leftrightarrow 3 \quad \ldots \quad Z \leftrightarrow 26 \quad AA \leftrightarrow 27 \ldots}$

Alice: Encryption

Represent the message $\mathcal{P}$ as an element of $\mathbb{Z}/M\mathbb{Z}$

(for example) $\boxed{A \leftrightarrow 1 \quad B \leftrightarrow 2 \quad C \leftrightarrow 3 \quad \ldots \quad Z \leftrightarrow 26 \quad AA \leftrightarrow 27 \ldots}$

$$\texttt{Sukumar} \leftrightarrow 19 \cdot 26^6 + 21 \cdot 26^5 + 11 \cdot 26^4 + 21 \cdot 26^3 + 12 \cdot 26^2 + 1 \cdot 26 + 18 = 6124312628$$

Note. Better if texts are not too short. Otherwise one performs some *padding*

Alice: Encryption

Represent the message $\mathcal{P}$ as an element of $\mathbb{Z}/M\mathbb{Z}$

(for example) $\boxed{A \leftrightarrow 1 \quad B \leftrightarrow 2 \quad C \leftrightarrow 3 \quad \ldots \quad Z \leftrightarrow 26 \quad AA \leftrightarrow 27 \ldots}$

`Sukumar` $\leftrightarrow 19 \cdot 26^6 + 21 \cdot 26^5 + 11 \cdot 26^4 + 21 \cdot 26^3 + 12 \cdot 26^2 + 1 \cdot 26 + 18 = 6124312628$

Note. Better if texts are not too short. Otherwise one performs some *padding*

$$\boxed{\mathcal{C} = E(\mathcal{P}) = \mathcal{P}^e \pmod{M}}$$

Alice: Encryption

Represent the message $\mathcal{P}$ as an element of $\mathbb{Z}/M\mathbb{Z}$

(for example) $\boxed{A \leftrightarrow 1 \quad B \leftrightarrow 2 \quad C \leftrightarrow 3 \quad \ldots \quad Z \leftrightarrow 26 \quad AA \leftrightarrow 27 \ldots}$

$$\texttt{Sukumar} \leftrightarrow 19 \cdot 26^6 + 21 \cdot 26^5 + 11 \cdot 26^4 + 21 \cdot 26^3 + 12 \cdot 26^2 + 1 \cdot 26 + 18 = 6124312628$$

Note. Better if texts are not too short. Otherwise one performs some *padding*

$$\boxed{\mathcal{C} = E(\mathcal{P}) = \mathcal{P}^e \pmod{M}}$$

Example: $p = 9049465727$, $q = 8789181607$, $M = 79537397720925283289$, $e = 2^{16} + 1 = 65537$,
$\mathcal{P} = \texttt{Sukumar}$:

Alice: Encryption

Represent the message $\mathcal{P}$ as an element of $\mathbb{Z}/M\mathbb{Z}$

(for example) $\boxed{A \leftrightarrow 1 \quad B \leftrightarrow 2 \quad C \leftrightarrow 3 \quad \dots \quad Z \leftrightarrow 26 \quad AA \leftrightarrow 27\dots}$

$$\texttt{Sukumar} \leftrightarrow 19 \cdot 26^6 + 21 \cdot 26^5 + 11 \cdot 26^4 + 21 \cdot 26^3 + 12 \cdot 26^2 + 1 \cdot 26 + 18 = 6124312628$$

Note. Better if texts are not too short. Otherwise one performs some *padding*

$$\boxed{\mathcal{C} = E(\mathcal{P}) = \mathcal{P}^e \pmod{M}}$$

Example: $p = 9049465727$, $q = 8789181607$, $M = 79537397720925283289$, $e = 2^{16} + 1 = 65537$, $\mathcal{P} = \texttt{Sukumar}$:

$$E(\texttt{Sukumar}) = 6124312628^{65537} \pmod{79537397720925283289}$$

$$= 25439695120356558116 = \mathcal{C} = \texttt{JGEBNBAUYTCOFJ}$$

Bob: Decryption

Bob: Decryption

$$\boxed{\mathcal{P} = D(\mathcal{C}) = \mathcal{C}^d \pmod{M}}$$

Bob: Decryption

$$\boxed{\mathcal{P} = D(\mathcal{C}) = \mathcal{C}^d \pmod{M}}$$

Note. Bob decrypts because he is the only one that knows d.

Bob: Decryption

$$\boxed{\mathcal{P} = D(\mathcal{C}) = \mathcal{C}^d \pmod{M}}$$

Note. Bob decrypts because he is the only one that knows d.

Theorem. (Euler) If $a, m \in \mathbb{N}$, $\gcd(a, m) = 1$,

$$a^{\varphi(m)} \equiv 1 \pmod{m}.$$

If $n_1 \equiv n_2 \bmod \varphi(m)$ then $a^{n_1} \equiv a^{n_2} \bmod m$.

Bob: Decryption

$$\boxed{\mathcal{P} = D(\mathcal{C}) = \mathcal{C}^d \pmod{M}}$$

Note. Bob decrypts because he is the only one that knows d.

> **Theorem. (Euler)** If $a, m \in \mathbb{N}$, $\gcd(a, m) = 1$,
> $$a^{\varphi(m)} \equiv 1 \pmod{m}.$$
> If $n_1 \equiv n_2 \bmod \varphi(m)$ then $a^{n_1} \equiv a^{n_2} \bmod m$.

Therefore $(ed \equiv 1 \bmod \varphi(M))$

$$\boxed{D(E(\mathcal{P})) = \mathcal{P}^{ed} \equiv \mathcal{P} \bmod M}$$

Bob: Decryption

$$\boxed{\mathcal{P} = D(\mathcal{C}) = \mathcal{C}^d \pmod{M}}$$

Note. Bob decrypts because he is the only one that knows d.

> **Theorem. (Euler)** If $a, m \in \mathbb{N}$, $\gcd(a, m) = 1$,
>
> $$a^{\varphi(m)} \equiv 1 \pmod{m}.$$
>
> If $n_1 \equiv n_2 \bmod \varphi(m)$ then $a^{n_1} \equiv a^{n_2} \bmod m$.

Therefore $(ed \equiv 1 \bmod \varphi(M))$

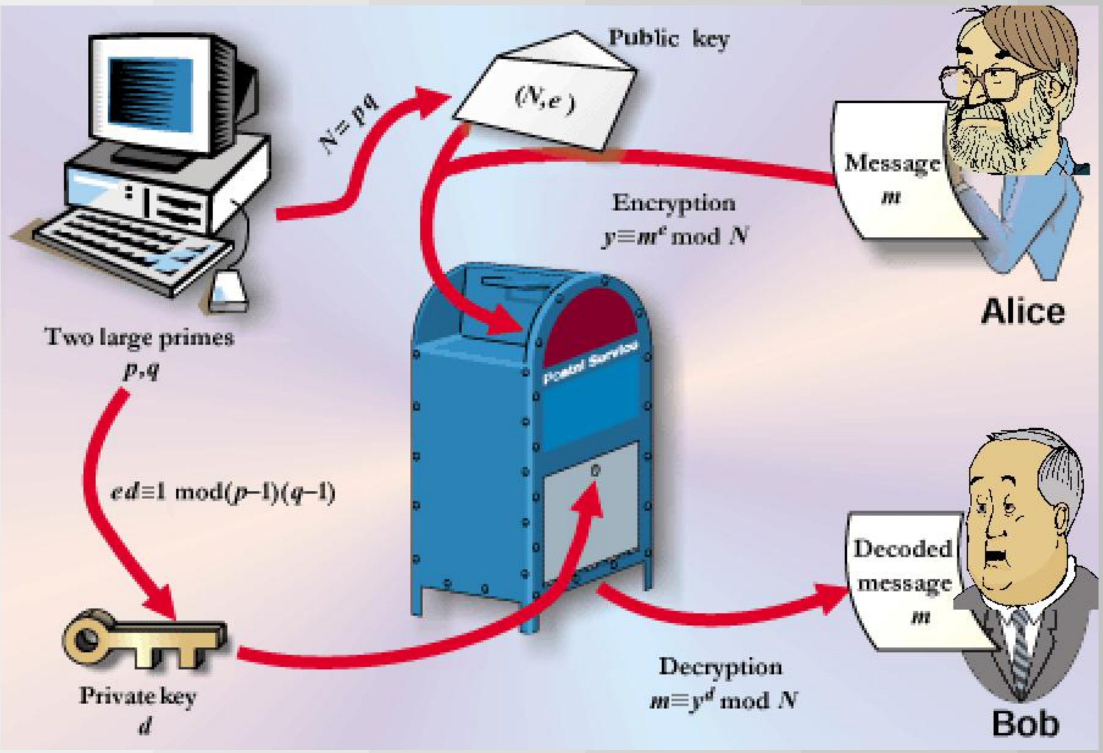
$$\boxed{D(E(\mathcal{P})) = \mathcal{P}^{ed} \equiv \mathcal{P} \bmod M}$$

Example(cont.):$d = 65537^{-1} \bmod \varphi(9049465727 \cdot 8789181607) = 57173914060643780153$

$D($`JGEBNBAUYTCOFJ`$) =$

$25439695120356558116^{57173914060643780153} (\mathrm{mod}\, 79537397720925283289) =$ `Sukumar`

RSA at work



Public key
(N,e)
N = pq
Message
m
Encryption
y≡m^e mod N
Alice
Two large primes
p,q
Postal Service
ed≡1 mod(p−1)(q−1)
Decoded
message
m
Private key
d
Decryption
m≡y^d mod N
Bob

Repeated squaring algorithm

Repeated squaring algorithm

Problem: How does one compute $a^b \bmod c$?

Repeated squaring algorithm

Problem: How does one compute $a^b \bmod c$?

$$2543969512035655811 6^{5717391406064378015 3} \pmod{795373977209252832 89}$$

Repeated squaring algorithm

Problem: How does one compute $a^b \bmod c$?

$$2543969512035655811 6^{5717391406064378015 3} \pmod{7953739772092528328 9}$$

✍

✍

✍

Repeated squaring algorithm

Problem: How does one compute $a^b \bmod c$?

$$2543969512035655811 6^{5717391406064378015 3} \pmod{7953739772092528328 9}$$

- Compute the binary expansion $b = \sum_{j=0}^{[\log_2 b]} \epsilon_j 2^j$
-
-

Repeated squaring algorithm

Problem: How does one compute $a^b \bmod c$?

$$2543969512035655811 6^{571739140606437801 53} \pmod{79537397720925283289}$$

- Compute the binary expansion $b = \sum_{j=0}^{[\log_2 b]} \epsilon_j 2^j$

  $571739140606437801 53 = 110001100101110010100010111110101011110011011000100100011000111001$

-

-

Repeated squaring algorithm

Problem: How does one compute $a^b \bmod c$?

$$25439695120356558116^{5717391406064378015 3} \pmod{795373977209252832 89}$$

- Compute the binary expansion $b = \sum_{j=0}^{[\log_2 b]} \epsilon_j 2^j$

  $5717391406064378015 3 = 110001100101110010100010111110101011100110110001001000110001 11001$

- Compute recursively $a^{2^j} \bmod c, j = 1, \ldots, [\log_2 b]$:

-

Repeated squaring algorithm

Problem: How does one compute $a^b \bmod c$?

$$2543969512035655811\,6^{5717391406064378015\,3} \pmod{7953739772092528328\,9}$$

✍ <u>Compute the binary expansion</u> $b = \sum_{j=0}^{[\log_2 b]} \epsilon_j 2^j$

5717391406064378015 3=1100011001011100101000101111101010111100110110001001000110001110 01

✍ <u>Compute recursively</u> $a^{2^j} \bmod c, j = 1, \ldots, [\log_2 b]$:

$$a^{2^j} \bmod c = \left(a^{2^{j-1}} \bmod c\right)^2 \bmod c$$

✍

Repeated squaring algorithm

Problem: How does one compute $a^b \bmod c$?

$$2543969512035655811 6^{5717391406064378015 3} \pmod{7953739772092528328 9}$$

- Compute the binary expansion $b = \sum_{j=0}^{[\log_2 b]} \epsilon_j 2^j$

$5717391406064378015 3 = 11000110010111001010001011111010101111001101100010010001100011100 1$

- Compute recursively $a^{2^j} \bmod c, j = 1, \ldots, [\log_2 b]$:

$$a^{2^j} \bmod c = \left(a^{2^{j-1}} \bmod c\right)^2 \bmod c$$

- Multiply the $a^{2^j} \bmod c$ with $\epsilon_j = 1$

Repeated squaring algorithm

Problem: How does one compute $a^b \bmod c$?

$$25439695120356558116^{57173914060643780153} \pmod{79537397720925283289}$$

- Compute the binary expansion $b = \sum_{j=0}^{[\log_2 b]} \epsilon_j 2^j$

$57173914060643780153 = 110001100101110010100010111110101011110011011000100100011000111001$

- Compute recursively $a^{2^j} \bmod c, j = 1, \ldots, [\log_2 b]$:

$$a^{2^j} \bmod c = \left(a^{2^{j-1}} \bmod c\right)^2 \bmod c$$

- Multiply the $a^{2^j} \bmod c$ with $\epsilon_j = 1$

$$a^b \bmod c = \left(\prod_{j=0, \epsilon_j=1}^{[\log_2 b]} a^{2^j} \bmod c\right) \bmod c$$

$$\#\{\textbf{oper. in } \mathbb{Z}/c\mathbb{Z} \textbf{ to compute } a^b \bmod c\} \leq 2\log_2 b$$

$\#\{\textbf{oper. in } \mathbb{Z}/c\mathbb{Z} \textbf{ to compute } a^b \bmod c\} \leq 2\log_2 b$

`JGEBNBAUYTCOFJ` is decrypted with 131 operations in

$$\mathbb{Z}/79537397720925283289\mathbb{Z}$$

$$\#\{\textbf{oper. in } \mathbb{Z}/c\mathbb{Z} \textbf{ to compute } a^b \bmod c\} \leq 2\log_2 b$$

`JGEBNBAUYTCOFJ` is decrypted with 131 operations in

$$\mathbb{Z}/795373977209252832 89\mathbb{Z}$$

PSEUDO CODE: $e_c(a,b) = a^b \bmod c$

$\#\{$**oper. in** $\mathbb{Z}/c\mathbb{Z}$ **to compute** $a^b \bmod c\} \leq 2\log_2 b$

`JGEBNBAUYTCOFJ` is decrypted with 131 operations in

$$\mathbb{Z}/795373977209252832 89\mathbb{Z}$$

PSEUDO CODE: $e_c(a,b) = a^b \bmod c$

$e_c(a,b)$	`=`	`if`	$b=1$	`then`	$a \bmod c$
		`if`	$2\mid b$	`then`	$e_c(a,\frac{b}{2})^2 \bmod c$
		`else`			$a * e_c(a,\frac{b-1}{2})^2 \bmod c$

$\boxed{\#\{\textbf{oper. in } \mathbb{Z}/c\mathbb{Z} \textbf{ to compute } a^b \bmod c\} \le 2\log_2 b}$

`JGEBNBAUYTCOFJ` is decrypted with 131 operations in

$$\mathbb{Z}/795373977209252832 89\mathbb{Z}$$

PSEUDO CODE: $e_c(a,b) = a^b \bmod c$

$e_c(a,b)$	`=`	`if`	$b=1$	`then`	$a \bmod c$
		`if`	$2 \mid b$	`then`	$e_c(a, \frac{b}{2})^2 \bmod c$
		`else`			$a * e_c(a, \frac{b-1}{2})^2 \bmod c$

To encrypt with $e = 2^{16}+1$, only 17 operations in $\mathbb{Z}/M\mathbb{Z}$ are enough

Key generation

Key generation

Problem. Produce a random prime $p \approx 10^{100}$

	Probabilistic algorithm (type Las Vegas)
1.	Let $p = \textsc{Random}(10^{100})$
2.	If isprime(p)=1 then Output=p else goto 1

Key generation

Problem. Produce a random prime $p \approx 10^{100}$

Probabilistic algorithm (type Las Vegas)
1. Let $p = \textsc{Random}(10^{100})$
2. If ISPRIME(p)=1 then OUTPUT=p else goto 1

subproblems:

Key generation

Problem. Produce a random prime $p \approx 10^{100}$

Probabilistic algorithm (type Las Vegas)
1. Let $p = \textsc{Random}(10^{100})$
2. If ISPRIME(p)=1 then OUTPUT=p else goto 1

subproblems:

A. How many iterations are necessary?

(i.e. how are primes distributes?)

Key generation

Problem. Produce a random prime $p \approx 10^{100}$

Probabilistic algorithm (type Las Vegas)	
1.	Let $p = \textsc{Random}(10^{100})$
2.	If ISPRIME(p)=1 then OUTPUT=p else goto 1

subproblems:

A. How many iterations are necessary?

(i.e. how are primes distributes?)

B. How does one check if p is prime?

(i.e. how does one compute ISPRIME(p)?) $\rightsquigarrow$ Primality test

Key generation

Problem. Produce a random prime $p \approx 10^{100}$

Probabilistic algorithm (type Las Vegas)	
1.	Let $p = \textsc{Random}(10^{100})$
2.	If ISPRIME(p)=1 then OUTPUT=p else goto 1

subproblems:

A. How many iterations are necessary?

(i.e. how are primes distributes?)

B. How does one check if p is prime?

(i.e. how does one compute ISPRIME(p)?) $\rightsquigarrow$ Primality test

False Metropolitan Legend: Check primality is equivalent to factoring

A. Distribution of prime numbers

A. Distribution of prime numbers

$$\pi(x) = \#\{p \leq x \text{ t. c. } p \text{ is prime}\}$$

A. Distribution of prime numbers

$$\pi(x) = \#\{p \leq x \text{ t. c. } p \text{ is prime}\}$$

Theorem. (Hadamard - de la vallee Pussen - 1897)

$$\pi(x) \sim \frac{x}{\log x}$$

A. Distribution of prime numbers

$$\pi(x) = \#\{p \leq x \text{ t. c. } p \text{ is prime}\}$$

Theorem. (Hadamard - de la vallee Pussen - 1897)

$$\pi(x) \sim \frac{x}{\log x}$$

Quantitative version:

Theorem. (Rosser - Schoenfeld) if $x \geq 67$

$$\frac{x}{\log x - 1/2} < \pi(x) < \frac{x}{\log x - 3/2}$$

A. Distribution of prime numbers

$$\pi(x) = \#\{p \leq x \text{ t. c. } p \text{ is prime}\}$$

Theorem. (Hadamard - de la vallee Pussen - 1897)

$$\pi(x) \sim \frac{x}{\log x}$$

Quantitative version:

Theorem. (Rosser - Schoenfeld) if $x \geq 67$

$$\frac{x}{\log x - 1/2} < \pi(x) < \frac{x}{\log x - 3/2}$$

Therefore

$$0.0043523959267 < Prob\left((\textsc{Random}(10^{100}) = \texttt{prime}\right) < 0.004371422086$$

If P_k is the probability that among k random numbers $\leq 10^{100}$ there is a prime one, then

If P_k is the probability that among k random numbers $\leq 10^{100}$ there is a prime one, then

$$P_k = 1 - \left(1 - \frac{\pi(10^{100})}{10^{100}}\right)^k$$

If P_k is the probability that among k random numbers $\leq 10^{100}$ there is a prime one, then

$$P_k = 1 - \left(1 - \frac{\pi(10^{100})}{10^{100}}\right)^k$$

Therefore

$$0.663942 < P_{250} < 0.66554440$$

If P_k is the probability that among k random numbers $\leq 10^{100}$ there is a prime one, then

$$P_k = 1 - \left(1 - \frac{\pi(10^{100})}{10^{100}}\right)^k$$

Therefore

$$0.663942 < P_{250} < 0.66554440$$

To speed up the process: One can consider only odd random numbers not divisible by 3 nor by 5.

If P_k is the probability that among k random numbers $\leq 10^{100}$ there is a prime one, then

$$P_k = 1 - \left(1 - \frac{\pi(10^{100})}{10^{100}}\right)^k$$

Therefore

$$0.663942 < P_{250} < 0.66554440$$

To speed up the process: One can consider only odd random numbers not divisible by 3 nor by 5.

Let

$$\Psi(x, 30) = \#\{n \leq x \text{ s.t.} \gcd(n, 30) = 1\}$$

To speed up the process: One can consider only odd random numbers not divisible by 3 nor by 5.

To speed up the process: One can consider only odd random numbers not divisible by 3 nor by 5.

Let

$$\Psi(x, 30) = \#\{n \leq x \text{ s.t. } \gcd(n, 30) = 1\}$$

then

To speed up the process: One can consider only odd random numbers not divisible by 3 nor by 5.

Let

$$\Psi(x, 30) = \#\{n \leq x \text{ s.t. } \gcd(n, 30) = 1\}$$

then

$$\frac{4}{15}x - 4 < \Psi(x, 30) < \frac{4}{15}x + 4$$

To speed up the process: One can consider only odd random numbers not divisible by 3 nor by 5.

Let

$$\Psi(x, 30) = \#\{n \leq x \text{ s.t. } \gcd(n, 30) = 1\}$$

then

$$\frac{4}{15}x - 4 < \Psi(x, 30) < \frac{4}{15}x + 4$$

Hence, if P'_k is the probability that among k random numbers $\leq 10^{100}$ coprime with 30, there is a prime one, then

To speed up the process: One can consider only odd random numbers not divisible by 3 nor by 5.

Let

$$\Psi(x, 30) = \#\{n \leq x \text{ s.t.} \gcd(n, 30) = 1\}$$

then

$$\frac{4}{15}x - 4 < \Psi(x, 30) < \frac{4}{15}x + 4$$

Hence, if P'_k is the probability that among k random numbers $\leq 10^{100}$ coprime with 30, there is a prime one, then

$$P'_k = 1 - \left(1 - \frac{\pi(10^{100})}{\Psi(10^{100}, 30)}\right)^k$$

To speed up the process: One can consider only odd random numbers not divisible by 3 nor by 5.

Let

$$\Psi(x, 30) = \#\{n \leq x \text{ s.t.} \gcd(n, 30) = 1\}$$

then

$$\frac{4}{15}x - 4 < \Psi(x, 30) < \frac{4}{15}x + 4$$

Hence, if P'_k is the probability that among k random numbers $\leq 10^{100}$ coprime with 30, there is a prime one, then

$$P'_k = 1 - \left(1 - \frac{\pi(10^{100})}{\Psi(10^{100}, 30)}\right)^k$$

and

$$0.98365832 < P'_{250} < 0.98395199$$

B. Primality test

B. Primality test

Fermat Little Theorem. If p is prime, $p \nmid a \in \mathbb{N}$

$$a^{p-1} \equiv 1 \bmod p$$

B. Primality test

Fermat Little Theorem. If p is prime, $p \nmid a \in \mathbb{N}$

$$a^{p-1} \equiv 1 \bmod p$$

NON-primality test

$$M \in \mathbb{Z}, \quad 2^{M-1} \not\equiv 1 \bmod M \Longrightarrow \quad M \text{composite!}$$

B. Primality test

Fermat Little Theorem. If p is prime, $p \nmid a \in \mathbb{N}$

$$a^{p-1} \equiv 1 \bmod p$$

NON-primality test

$$M \in \mathbb{Z}, \quad 2^{M-1} \not\equiv 1 \bmod M \implies \quad M \text{composite!}$$

EXAMPLE: $2^{RSA_{2048}-1} \not\equiv 1 \bmod RSA_{2048}$

Therefore RSA_{2048} is composite!

B. Primality test

Fermat Little Theorem. If p is prime, $p \nmid a \in \mathbb{N}$

$$a^{p-1} \equiv 1 \bmod p$$

NON-primality test

$$M \in \mathbb{Z}, \quad 2^{M-1} \not\equiv 1 \bmod M \Longrightarrow \quad M \text{composite!}$$

EXAMPLE: $2^{RSA_{2048}-1} \not\equiv 1 \bmod RSA_{2048}$

Therefore RSA_{2048} is composite!

Fermat little Theorem does not invert. Infact

B. Primality test

Fermat Little Theorem. If p is prime, $p \nmid a \in \mathbb{N}$

$$a^{p-1} \equiv 1 \bmod p$$

NON-primality test

$$M \in \mathbb{Z}, \quad 2^{M-1} \not\equiv 1 \bmod M \implies \quad M \text{composite!}$$

EXAMPLE: $2^{RSA_{2048}-1} \not\equiv 1 \bmod RSA_{2048}$

Therefore RSA_{2048} is composite!

Fermat little Theorem does not invert. Infact

$$2^{93960} \equiv 1 \pmod{93961} \quad \text{but} \quad 93961 = 7 \times 31 \times 433$$

Strong pseudo primes

Strong pseudo primes

From now on $m \equiv 3 \bmod 4$ (just to simplify the notation)

Strong pseudo primes

From now on $m \equiv 3 \bmod 4$ (just to simplify the notation)

Definition. $m \in \mathbb{N}$, $m \equiv 3 \bmod 4$, composite is said strong pseudo prime (SPSP) in base a if

$$a^{(m-1)/2} \equiv \pm 1 \pmod{m}.$$

Strong pseudo primes

From now on $m \equiv 3 \bmod 4$ (just to simplify the notation)

Definition. $m \in \mathbb{N}$, $m \equiv 3 \bmod 4$, composite is said strong pseudo prime (SPSP) in base a if

$$a^{(m-1)/2} \equiv \pm 1 \pmod{m}.$$

Note. If $p > 2$ prime $\Longrightarrow$ $a^{(p-1)/2} \equiv \pm 1 \pmod{p}$

Let $\boxed{\mathcal{S} = \{a \in \mathbb{Z}/m\mathbb{Z} \text{ s.t. } \gcd(m, a) = 1, a^{(m-1)/2} \equiv \pm 1 \pmod{m}\}}$

Strong pseudo primes

From now on $m \equiv 3 \bmod 4$ (just to simplify the notation)

Definition. $m \in \mathbb{N}$, $m \equiv 3 \bmod 4$, composite is said strong pseudo prime (SPSP) in base a if

$$a^{(m-1)/2} \equiv \pm 1 \pmod{m}.$$

Note. If $p > 2$ prime $\Longrightarrow \quad a^{(p-1)/2} \equiv \pm 1 \pmod{p}$

Let $\boxed{\mathcal{S} = \{a \in \mathbb{Z}/m\mathbb{Z} \text{ s.t. } \gcd(m, a) = 1, a^{(m-1)/2} \equiv \pm 1 \pmod{m}\}}$

①

②

③

④

Strong pseudo primes

From now on $m \equiv 3 \bmod 4$ (just to simplify the notation)

Definition. $m \in \mathbb{N}$, $m \equiv 3 \bmod 4$, composite is said strong pseudo prime (SPSP) in base a if

$$a^{(m-1)/2} \equiv \pm 1 \pmod{m}.$$

Note. If $p > 2$ prime $\Longrightarrow$ $a^{(p-1)/2} \equiv \pm 1 \pmod{p}$

Let $\boxed{\mathcal{S} = \{a \in \mathbb{Z}/m\mathbb{Z} \text{ s.t. } \gcd(m,a) = 1, a^{(m-1)/2} \equiv \pm 1 \pmod{m}\}}$

① $\mathcal{S} \subseteq (\mathbb{Z}/m\mathbb{Z})^*$ subgroup

②

③

④

Strong pseudo primes

From now on $m \equiv 3 \bmod 4$ (just to simplify the notation)

Definition. $m \in \mathbb{N}$, $m \equiv 3 \bmod 4$, composite is said strong pseudo prime (SPSP) in base a if

$$a^{(m-1)/2} \equiv \pm 1 \pmod{m}.$$

Note. If $p > 2$ prime $\Longrightarrow$ $a^{(p-1)/2} \equiv \pm 1 \pmod{p}$

Let $\boxed{\mathcal{S} = \{a \in \mathbb{Z}/m\mathbb{Z} \text{ s.t. } \gcd(m, a) = 1, a^{(m-1)/2} \equiv \pm 1 \pmod{m}\}}$

① $\mathcal{S} \subseteq (\mathbb{Z}/m\mathbb{Z})^*$ subgroup

② If m is composite $\Longrightarrow$ proper subgroup

③

④

Strong pseudo primes

From now on $m \equiv 3 \bmod 4$ (just to simplify the notation)

Definition. $m \in \mathbb{N}$, $m \equiv 3 \bmod 4$, composite is said strong pseudo prime (SPSP) in base a if

$$a^{(m-1)/2} \equiv \pm 1 \pmod{m}.$$

Note. If $p > 2$ prime $\Longrightarrow$ $a^{(p-1)/2} \equiv \pm 1 \pmod{p}$

Let $\boxed{\mathcal{S} = \{a \in \mathbb{Z}/m\mathbb{Z} \text{ s.t. } \gcd(m,a) = 1, a^{(m-1)/2} \equiv \pm 1 \pmod{m}\}}$

① $\mathcal{S} \subseteq (\mathbb{Z}/m\mathbb{Z})^*$ subgroup

② If m is composite $\Longrightarrow$ proper subgroup

③ If m is composite $\Longrightarrow$ $\#\mathcal{S} \leq \frac{\varphi(m)}{4}$

④

Strong pseudo primes

From now on $m \equiv 3 \bmod 4$ (just to simplify the notation)

Definition. $m \in \mathbb{N}$, $m \equiv 3 \bmod 4$, composite is said strong pseudo prime (SPSP) in base a if

$$a^{(m-1)/2} \equiv \pm 1 \pmod{m}.$$

Note. If $p > 2$ prime $\Longrightarrow$ $a^{(p-1)/2} \equiv \pm 1 \pmod{p}$

Let $\boxed{\mathcal{S} = \{a \in \mathbb{Z}/m\mathbb{Z} \text{ s.t. } \gcd(m,a) = 1, a^{(m-1)/2} \equiv \pm 1 \pmod{m}\}}$

① $\mathcal{S} \subseteq (\mathbb{Z}/m\mathbb{Z})^*$ subgroup

② If m is composite $\Longrightarrow$ proper subgroup

③ If m is composite $\Longrightarrow$ $\#\mathcal{S} \leq \frac{\varphi(m)}{4}$

④ If m is composite $\Longrightarrow$ $Prob(m \text{ PSPF in base } a) \leq 0,25$

Miller–Rabin primality test

Miller–Rabin primality test

Let $m \equiv 3 \bmod 4$

Miller–Rabin primality test

Let $m \equiv 3 \bmod 4$

MILLER RABIN ALGORITHM WITH k ITERATIONS
$N = (m-1)/2$
`for` $j = 0$ `to` k `do` $a =$`Random`(m)
`if` $a^N \not\equiv \pm 1 \bmod m$ `then OUPUT=(`m `composite): END`
`endfor OUTPUT=(`m `prime)`

Miller–Rabin primality test

Let $m \equiv 3 \bmod 4$

MILLER RABIN ALGORITHM WITH k ITERATIONS
$N = (m-1)/2$
`for` $j = 0$ `to` k `do` $a =$`Random`(m)
`if` $a^N \not\equiv \pm 1 \bmod m$ `then OUPUT=(`m `composite): END`
`endfor OUTPUT=(`m `prime)`

Monte Carlo primality test

Miller–Rabin primality test

Let $m \equiv 3 \bmod 4$

MILLER RABIN ALGORITHM WITH k ITERATIONS
$N = (m-1)/2$
`for` $j = 0$ `to` k `do` $a =$`Random`(m)
`if` $a^N \not\equiv \pm 1 \bmod m$ `then OUPUT=(`m `composite): END`
`endfor OUTPUT=(`m `prime)`

Monte Carlo primality test

$Prob$(Miller Rabin says m `prime` and m is composite) $\lessapprox \frac{1}{4^k}$

Miller–Rabin primality test

Let $m \equiv 3 \bmod 4$

MILLER RABIN ALGORITHM WITH k ITERATIONS
$N = (m-1)/2$
`for` $j = 0$ `to` k `do` $a =$`Random`(m)
`if` $a^N \not\equiv \pm 1 \bmod m$ `then OUPUT=(`m `composite): END`
`endfor OUTPUT=(`m `prime)`

Monte Carlo primality test

$Prob$(Miller Rabin says m `prime` and m is composite) $\lessapprox \frac{1}{4^k}$

In the real world, software uses Miller Rabin with $k = 10$

Deterministic primality tests

Deterministic primality tests

Theorem. (Miller, Bach) If m is composite, then

$$\textbf{GRH} \Rightarrow \exists a \leq 2\log^2 m \text{ s.t. } a^{(m-1)/2} \not\equiv \pm 1 \pmod{m}.$$

(i.e. *m is not SPSP in base a.*)

Deterministic primality tests

Theorem. (Miller, Bach) If m is composite, then

$$\textbf{GRH} \Rightarrow \ \exists a \leq 2\log^2 m \text{ s.t. } \ a^{(m-1)/2} \not\equiv \pm 1 \pmod{m}.$$

(i.e. *m is not SPSP in base a.*)

Consequence: "Miller–Rabin de–randomizes on GRH" ($m \equiv 3 \bmod 4$)

Deterministic primality tests

Theorem. (Miller, Bach) If m is composite, then

$$\mathbf{GRH} \Rrightarrow \; \exists a \le 2\log^2 m \text{ s.t. } \; a^{(m-1)/2} \not\equiv \pm 1 \pmod{m}.$$

(i.e. *m is not SPSP in base a.*)

Consequence: "Miller–Rabin de–randomizes on GRH" ($m \equiv 3 \bmod 4$)

`for`	$a = 2$ `to` $2\log^2 m$	`do`	
	`if` $a^{(m-1)/2} \not\equiv \pm 1 \bmod m$	`then`	
		`OUPUT=(`m `composite):`	`END`
`endfor`		`OUTPUT=(`m `prime)`	

Deterministic primality tests

Theorem. (Miller, Bach) If m is composite, then

$$\mathbf{GRH} \Rightarrow \exists a \leq 2\log^2 m \text{ s.t. } a^{(m-1)/2} \not\equiv \pm 1 \pmod{m}.$$

(i.e. *m is not SPSP in base a.*)

Consequence: "Miller–Rabin de–randomizes on GRH" ($m \equiv 3 \bmod 4$)

`for`	$a = 2$ `to` $2\log^2 m$	`do`	
	`if` $a^{(m-1)/2} \not\equiv \pm 1 \bmod m$	`then`	
		`OUPUT=(`m `composite):`	`END`
`endfor`		`OUTPUT=(`m `prime)`	

Deterministic Polynomial time algorithm

Deterministic primality tests

Theorem. (Miller, Bach) If m is composite, then

$$\textbf{GRH} \Rightarrow \ \exists a \leq 2\log^2 m \text{ s.t. } \ a^{(m-1)/2} \not\equiv \pm 1 \pmod{m}.$$

(i.e. *m is not SPSP in base a.*)

Consequence: "Miller–Rabin de–randomizes on GRH" ($m \equiv 3 \bmod 4$)

`for`	$a = 2$ `to` $2\log^2 m$	`do`	
	`if` $a^{(m-1)/2} \not\equiv \pm 1 \bmod m$	`then`	
		`OUPUT=(`m `composite):`	`END`
`endfor`		`OUTPUT=(`m `prime)`	

Deterministic Polynomial time algorithm

It runs in $O(\log^5 m)$ operations in $\mathbb{Z}/m\mathbb{Z}$.

Certified prime records

Certified prime records

- ✎
- ✎
- ✎
- ✎
- ✎
- ✎
- ✎
- ✎

Certified prime records

- $2^{20996011} - 1$, 6320430 digits (discovered in 2003)
-
-
-
-
-
-
-

Certified prime records

- $2^{20996011} - 1$, 6320430 digits (discovered in 2003)
- $2^{13466917} - 1$, 4053946 digits (discovered in 2001)
-
-
-
-
-
-

Certified prime records

- $2^{20996011} - 1$, 6320430 digits (discovered in 2003)
- $2^{13466917} - 1$, 4053946 digits (discovered in 2001)
- $2^{6972593} - 1$, 2098960 digits (discovered in 1999)

Certified prime records

- $2^{20996011} - 1$, 6320430 digits (discovered in 2003)
- $2^{13466917} - 1$, 4053946 digits (discovered in 2001)
- $2^{6972593} - 1$, 2098960 digits (discovered in 1999)
- $5359 \times 2^{5054502} + 1$, 1521561 digits (discovered in 2003)
-
-
-
-

Certified prime records

- $2^{20996011} - 1$, 6320430 digits (discovered in 2003)
- $2^{13466917} - 1$, 4053946 digits (discovered in 2001)
- $2^{6972593} - 1$, 2098960 digits (discovered in 1999)
- $5359 \times 2^{5054502} + 1$, 1521561 digits (discovered in 2003)
- $2^{3021377} - 1$, 909526 digits (discovered in 1998)

Certified prime records

- $2^{20996011} - 1$, 6320430 digits (discovered in 2003)
- $2^{13466917} - 1$, 4053946 digits (discovered in 2001)
- $2^{6972593} - 1$, 2098960 digits (discovered in 1999)
- $5359 \times 2^{5054502} + 1$, 1521561 digits (discovered in 2003)
- $2^{3021377} - 1$, 909526 digits (discovered in 1998)
- $2^{2976221} - 1$, 895932 digits (discovered in 1997)
-
-

Certified prime records

- $2^{20996011} - 1$, 6320430 digits (discovered in 2003)
- $2^{13466917} - 1$, 4053946 digits (discovered in 2001)
- $2^{6972593} - 1$, 2098960 digits (discovered in 1999)
- $5359 \times 2^{5054502} + 1$, 1521561 digits (discovered in 2003)
- $2^{3021377} - 1$, 909526 digits (discovered in 1998)
- $2^{2976221} - 1$, 895932 digits (discovered in 1997)
- $1372930^{131072} + 1$, 804474 digits (discovered in 2003)
-

Certified prime records

- $2^{20996011} - 1$, 6320430 digits (discovered in 2003)
- $2^{13466917} - 1$, 4053946 digits (discovered in 2001)
- $2^{6972593} - 1$, 2098960 digits (discovered in 1999)
- $5359 \times 2^{5054502} + 1$, 1521561 digits (discovered in 2003)
- $2^{3021377} - 1$, 909526 digits (discovered in 1998)
- $2^{2976221} - 1$, 895932 digits (discovered in 1997)
- $1372930^{131072} + 1$, 804474 digits (discovered in 2003)
- $1176694^{131072} + 1$, 795695 digits (discovered in 2003)

The AKS deterministic primality test

The AKS deterministic primality test

Department of Computer Science & Engineering,
I.I.T. Kanpur, Agost 8, 2002.

The AKS deterministic primality test

Department of Computer Science & Engineering,
I.I.T. Kanpur, Agost 8, 2002.

Nitin Saxena, Neeraj Kayal and Manindra Agarwal

The AKS deterministic primality test

Department of Computer Science & Engineering,
I.I.T. Kanpur, Agost 8, 2002.

Nitin Saxena, Neeraj Kayal and Manindra Agarwal
New deterministic, polynomial–time, primality test.

The AKS deterministic primality test

Department of Computer Science & Engineering,
I.I.T. Kanpur, Agost 8, 2002.

Nitin Saxena, Neeraj Kayal and Manindra Agarwal

New deterministic, polynomial–time, primality test.

Solves #1 open question in computational number theory

The AKS deterministic primality test

Department of Computer Science & Engineering,
I.I.T. Kanpur, Agost 8, 2002.

Nitin Saxena, Neeraj Kayal and Manindra Agarwal

New deterministic, polynomial–time, primality test.

Solves #1 open question in computational number theory

http://www.cse.iitk.ac.in/news/primality.html

How does the AKS work?

How does the AKS work?

Theorem. (AKS) Let $n \in \mathbb{N}$. Assume q, r primes, $S \subseteq \mathbb{N}$ finite:

- $q|r-1$;
- $n^{(r-1)/q} \bmod r \notin \{0, 1\}$;
- $\gcd(n, b-b') = 1, \quad \forall b, b' \in S$ (distinct);
- $\binom{q+\#S-1}{\#S} \geq n^{2\lfloor\sqrt{r}\rfloor}$;
- $(x+b)^n = x^n + b$ in $\mathbb{Z}/n\mathbb{Z}[x]/(x^r-1), \quad \forall b \in S$;

Then n is a power of a prime

Bernstein formulation

How does the AKS work?

Theorem. (AKS) Let $n \in \mathbb{N}$. Assume q, r primes, $S \subseteq \mathbb{N}$ finite:

- $q | r - 1$;
- $n^{(r-1)/q} \bmod r \notin \{0, 1\}$;
- $\gcd(n, b - b') = 1, \quad \forall b, b' \in S$ (distinct);
- $\binom{q+\#S-1}{\#S} \geq n^{2\lfloor\sqrt{r}\rfloor}$;
- $(x + b)^n = x^n + b$ in $\mathbb{Z}/n\mathbb{Z}[x]/(x^r - 1), \quad \forall b \in S$;

Then n is a power of a prime Bernstein formulation

Fouvry Theorem (1985) $\Rightarrow \quad \exists r \approx \log^6 n, s \approx \log^4 n$

How does the AKS work?

Theorem. (AKS) Let $n \in \mathbb{N}$. Assume q, r primes, $S \subseteq \mathbb{N}$ finite:

- $q | r - 1$;
- $n^{(r-1)/q} \bmod r \notin \{0, 1\}$;
- $\gcd(n, b - b') = 1, \quad \forall b, b' \in S$ (distinct);
- $\binom{q+\#S-1}{\#S} \geq n^{2\lfloor\sqrt{r}\rfloor}$;
- $(x + b)^n = x^n + b$ in $\mathbb{Z}/n\mathbb{Z}[x]/(x^r - 1), \quad \forall b \in S$;

Then n is a power of a prime   Bernstein formulation

Fouvry Theorem (1985) $\Longrightarrow \quad \exists r \approx \log^6 n, s \approx \log^4 n$

$\Longrightarrow$ AKS runs in $O(\log^{17} n)$ operations in $\mathbb{Z}/n\mathbb{Z}$.

How does the AKS work?

Theorem. (AKS) Let $n \in \mathbb{N}$. Assume q, r primes, $S \subseteq \mathbb{N}$ finite:

- $q | r - 1$;
- $n^{(r-1)/q} \bmod r \notin \{0, 1\}$;
- $\gcd(n, b - b') = 1, \quad \forall b, b' \in S$ (distinct);
- $\binom{q+\#S-1}{\#S} \geq n^{2\lfloor\sqrt{r}\rfloor}$;
- $(x + b)^n = x^n + b$ in $\mathbb{Z}/n\mathbb{Z}[x]/(x^r - 1), \quad \forall b \in S$;

Then n is a power of a prime Bernstein formulation

Fouvry Theorem (1985) $\Longrightarrow \quad \exists r \approx \log^6 n, s \approx \log^4 n$

$\Longrightarrow$ AKS runs in $O(\log^{17} n)$ operations in $\mathbb{Z}/n\mathbb{Z}$.

Many simplifications and improvements: Bernstein, Lenstra, Pomerance.....

Why is RSA safe?

Why is RSA safe?

☞

☞

☞

Why is RSA safe?

- It is clear that if Charles can factor M,
-
-

Why is RSA safe?

- It is clear that if Charles can factor M,
  then he can also compute $\varphi(M)$ and then also d so to decrypt messages
-
-

Why is RSA safe?

- It is clear that if Charles can factor M,
  then he can also compute $\varphi(M)$ and then also d so to decrypt messages
- Computing $\varphi(M)$ is equivalent to completely factor M. In fact
-

Why is RSA safe?

- It is clear that if Charles can factor M,
  then he can also compute $\varphi(M)$ and then also d so to decrypt messages
- Computing $\varphi(M)$ is equivalent to completely factor M. In fact

$$p, q = \frac{M - \varphi(M) + 1 \pm \sqrt{(M - \varphi(M) + 1)^2 - 4M}}{2}$$

-

Why is RSA safe?

- ☞ It is clear that if Charles can factor M,
  then he can also compute $\varphi(M)$ and then also d so to decrypt messages
- ☞ Computing $\varphi(M)$ is equivalent to completely factor M. In fact

$$p, q = \frac{M - \varphi(M) + 1 \pm \sqrt{(M - \varphi(M) + 1)^2 - 4M}}{2}$$

- ☞ **RSA Hypothesis.** The only way to compute efficiently

Why is RSA safe?

- It is clear that if Charles can factor M,
  then he can also compute $\varphi(M)$ and then also d so to decrypt messages
- Computing $\varphi(M)$ is equivalent to completely factor M. In fact

$$p, q = \frac{M - \varphi(M) + 1 \pm \sqrt{(M - \varphi(M) + 1)^2 - 4M}}{2}$$

- **RSA Hypothesis.** The only way to compute efficiently

$$x^{1/e} \bmod M, \quad \forall x \in \mathbb{Z}/M\mathbb{Z}$$

Why is RSA safe?

- It is clear that if Charles can factor M,
  then he can also compute $\varphi(M)$ and then also d so to decrypt messages
- Computing $\varphi(M)$ is equivalent to completely factor M. In fact

$$p, q = \frac{M - \varphi(M) + 1 \pm \sqrt{(M - \varphi(M) + 1)^2 - 4M}}{2}$$

- **RSA Hypothesis.** The only way to compute efficiently

$$x^{1/e} \bmod M, \quad \forall x \in \mathbb{Z}/M\mathbb{Z}$$

  (i.e. decrypt messages) is to factor M

Why is RSA safe?

☞ It is clear that if Charles can factor M,
then he can also compute $\varphi(M)$ and then also d so to decrypt messages

☞ Computing $\varphi(M)$ is equivalent to completely factor M. In fact

$$p, q = \frac{M - \varphi(M) + 1 \pm \sqrt{(M - \varphi(M) + 1)^2 - 4M}}{2}$$

☞ **RSA Hypothesis.** The only way to compute efficiently

$$x^{1/e} \bmod M, \quad \forall x \in \mathbb{Z}/M\mathbb{Z}$$

(i.e. decrypt messages) is to factor M

In other words

Why is RSA safe?

- It is clear that if Charles can factor M,
  then he can also compute $\varphi(M)$ and then also d so to decrypt messages
- Computing $\varphi(M)$ is equivalent to completely factor M. In fact

$$p, q = \frac{M - \varphi(M) + 1 \pm \sqrt{(M - \varphi(M) + 1)^2 - 4M}}{2}$$

- **RSA Hypothesis.** The only way to compute efficiently

  $x^{1/e} \bmod M, \quad \forall x \in \mathbb{Z}/M\mathbb{Z}$

  (i.e. decrypt messages) is to factor M

  In other words

  The two problems are polynomially equivalent

Two kinds of Cryptography

Two kinds of Cryptography

- **Private key (or symmetric)**
  - Lucifer
  - DES
  - AES

Two kinds of Cryptography

- **Private key (or symmetric)**
  - Lucifer
  - DES
  - AES
- **Public key**
  - RSA
  - Diffie–Hellmann
  - Knapsack
  - NTRU