Entwicklung der Asymmetrischen Kryptographie und deren Einsatz

Peter Kraml, 5a hlw

Facharbeit Mathematik

Schuljahr 2013/14

Caesar-Verschlüsselung

- Verschiebung der Buchstaben im Alphabet
- sehr leicht zu Entschlüsseln

Beispiel

```
Ausgang:  ABCDEFGHIJKLMNOPQRSTUVWXYZ
Caesar:   DEFGHIJKLMNOPQRSTUVWXYZABC
```

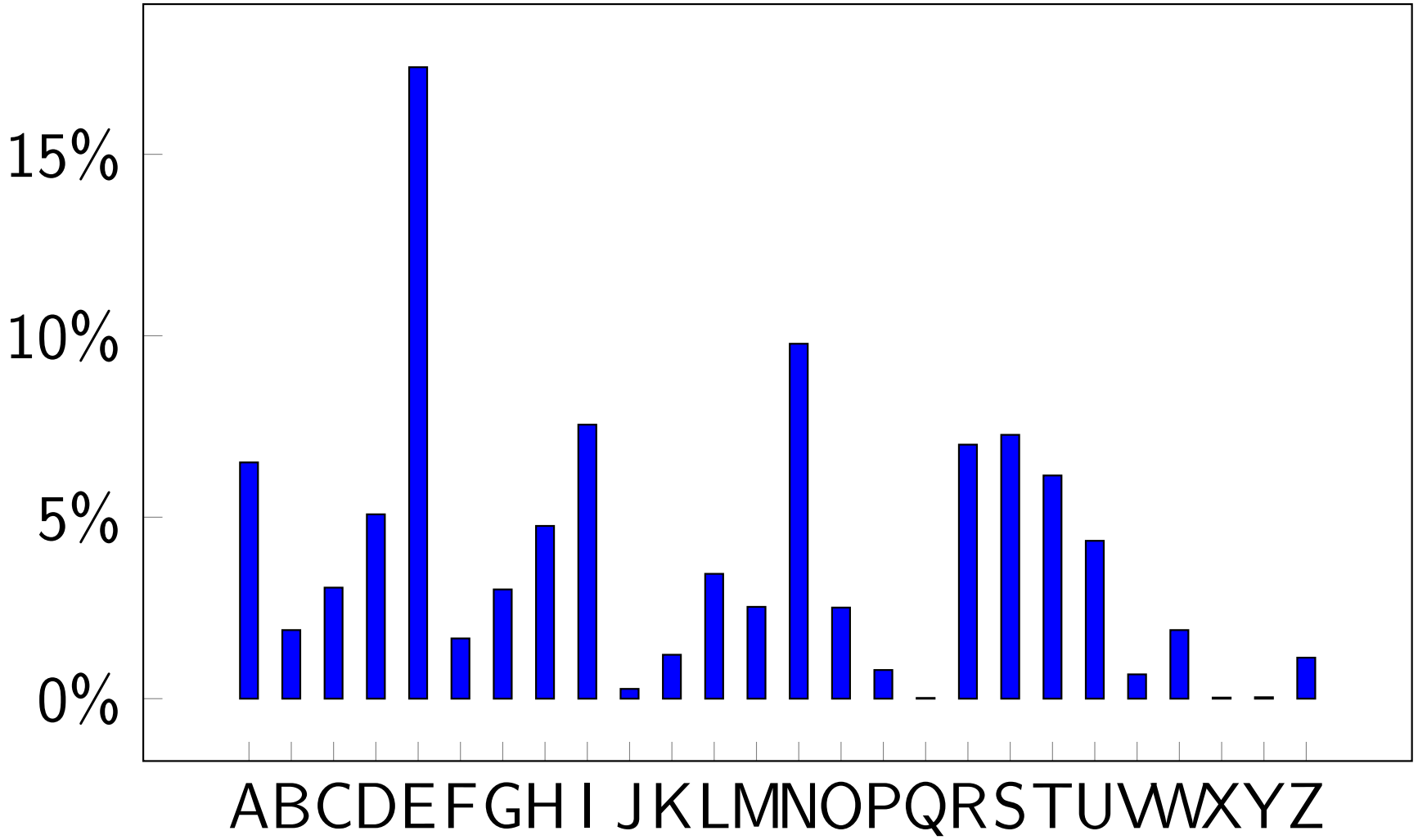
Erweiterung

- Verwendung von vermischten Alphabet
- ergibt $26! \cong 4 * 10^{26}$ Möglichkeiten

Beispiel

```
Ausgang:  ABCDEFGHIJKLMNOPQRSTUVWXYZ
Caesar:   DTURJKCBGPLMNEFQIOSYHVWXZA
```

Buchstabenhäufigkeit Deutsch



Schlüsselaustausch

- Schlüsselaustausch von Sender zu Empfänger
- Angreifer könnte Schlüssel abhören
- Sicherste Möglichkeit: persönlicher Austausch

Diffie-Hellman-Schlüsselaustausch

- Martin Hellman, Whitfield Diffie, Ralph Merkle
- 1976, Stanford-Universität, Kalifornien

Abbildung : V. l. n. r.: Ralph Merkle, Martin Hellman, Whitfield Diffie

Diffie-Hellman-Schlüsselaustausch

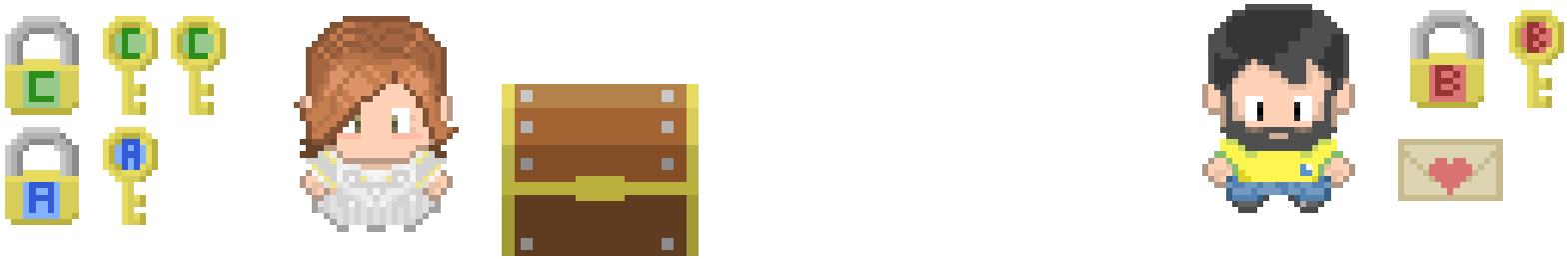


Abbildung : Bob möchte Alice einen Liebesbrief senden

Diffie-Hellman-Schlüsselaustausch

Abbildung : Alice legt ihr Schloss C in eine Kiste

Diffie-Hellman-Schlüsselaustausch

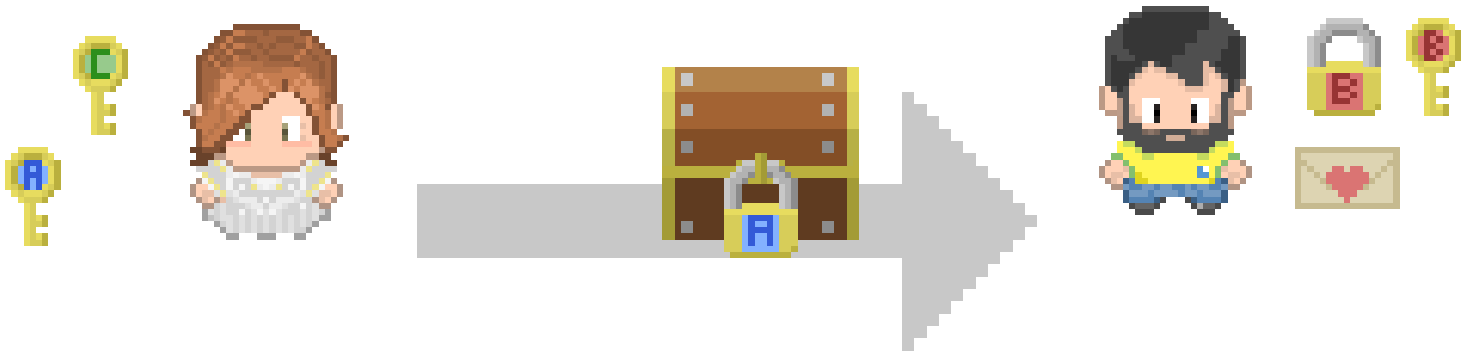


Abbildung : Alice versperrt die Kiste mit ihrem Schloss A und schickt sie an Bob

Diffie-Hellman-Schlüsselaustausch

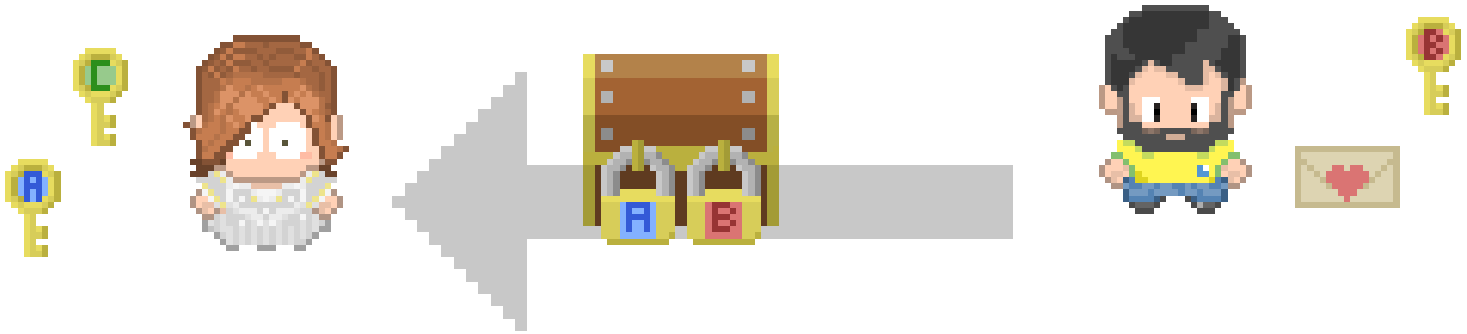


Abbildung : Bob versperrt die Kiste zusätzlich mit seinem Schloss B und sendet sie zurück

Diffie-Hellman-Schlüsselaustausch

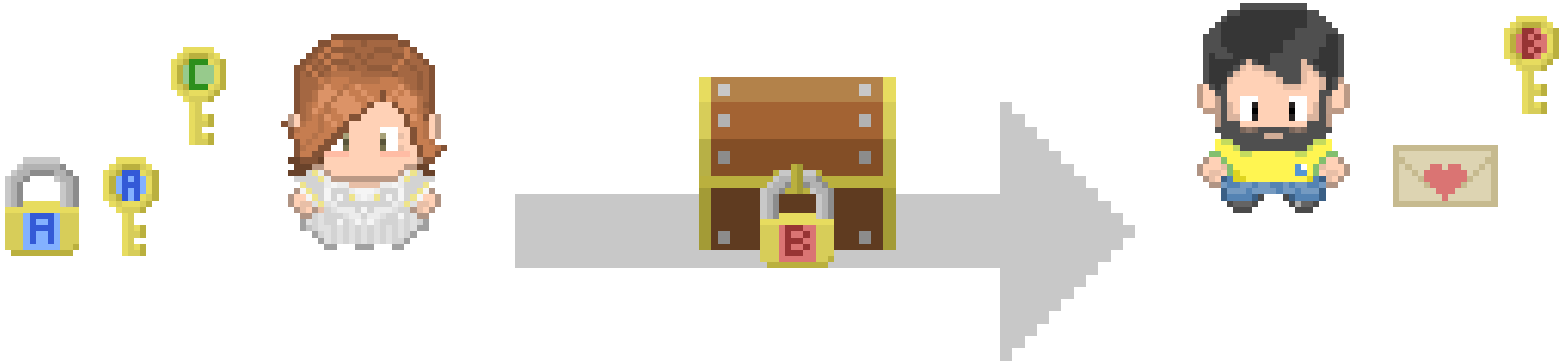


Abbildung : Alice nimmt ihr Schloss A wieder von der Kiste ab und sendet die Kiste erneut an Bob

Diffie-Hellman-Schlüsselaustausch

Abbildung : Bob kann die Kiste mit Schlüssel B öffnen und seinen Liebesbrief mit Schlüssel und Schloss C versperren

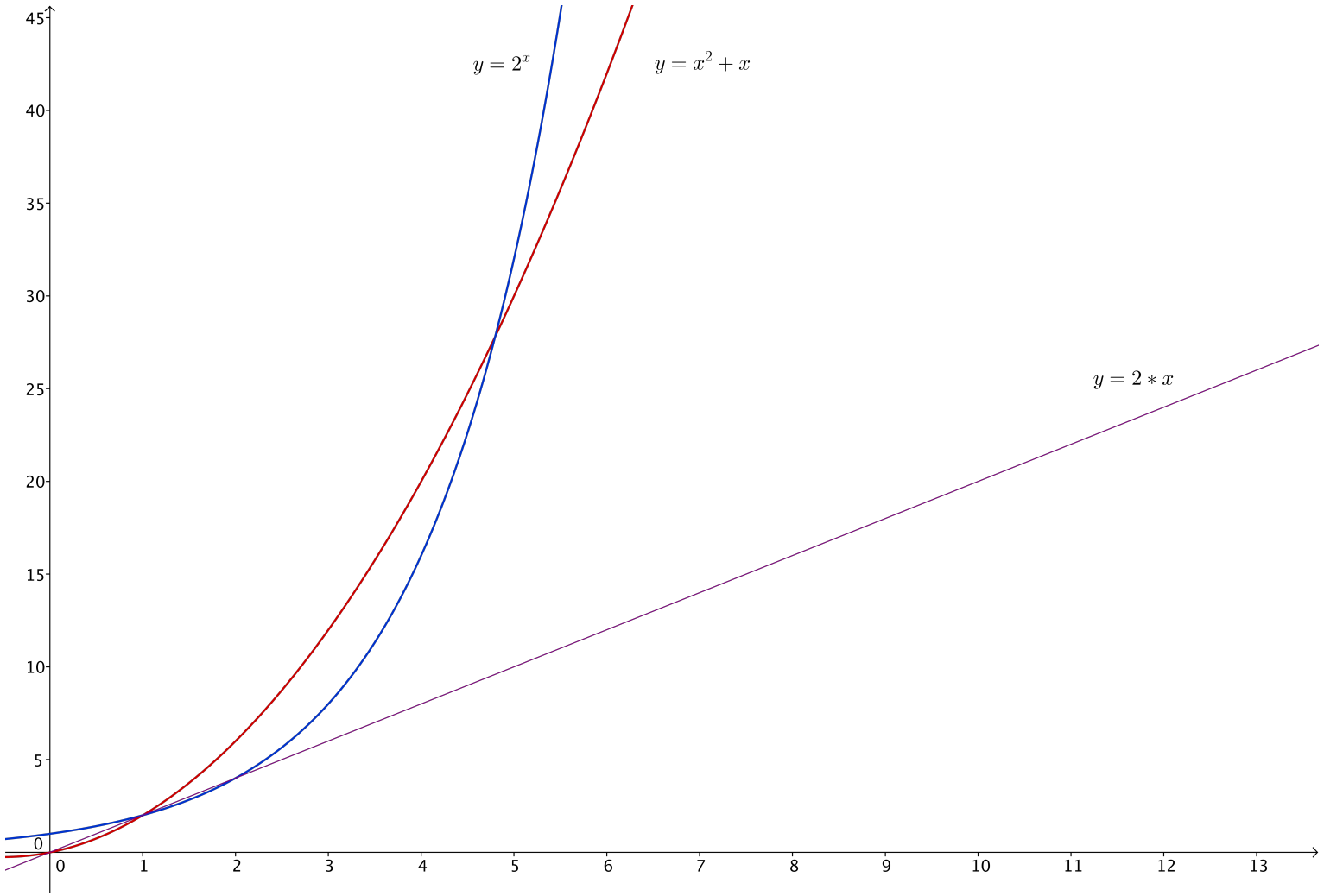
Einwegfunktionen

- mathematische Funktion
- »leicht« in eine Richtung berechenbar
- »schwer« in die entgegengesetzte Richtung berechenbar
- Definition durch Polynomialzeit

Polynomialzeit

- Grenze zwischen »schwer« und »leicht«
- benötigte Zeit darf nicht größer als polynomial zur Problemgröße steigen

Polynomialzeit



Polynomialzeit

- benötigte Zeit: (z. B. Rechenoperationen)
- Problemgröße: Anzahl der Bits im Schlüssel
- hängt von Algorithmus ab

RSA-Verfahren

- Rivest, Shamir, Adleman
- asymmetrisches Verschlüsselungsverfahren
- 1977 am MIT in Kalifornien entwickelt

Die Erfinder

Abbildung : V. l. n. r.: Shamir, Rivest und Adleman

Die Nachrichtendienste

- Government Communications Headquarter (GCHQ)
- sehr ähnliches Verfahren
- Mitte der 1960er Jahre
- musste jedoch geheim gehalten werden

Schlüsselgenerierung

- Privater Schlüssel $\{d, N\}$
- Öffentlicher Schlüssel $\{e, N\}$
- N für beide Schlüssel ident

1. Schritt: Primzahlen

- 2 Primzahlen p und q
- dürfen nicht gleich sein

Beispiel

$p = 3$
$q = 7$

2. Schritt: RSA-Modul

- N ist Produkt von p mal q

Beispiel

$N = p * q$
$N = 3 * 7$
$N = 21$

3. Schritt: Eulersche-Φ-Funktion

- $\Phi()$: wie viele kleinere Zahlen auch teilerfremd sind
- es gilt: $\Phi(x) = x - 1$ wenn x eine Primzahl ist
- und: $\Phi(a * b) = \Phi(a) * \Phi(b)$
- wird von RSA-Modul berechnet

3. Schritt: Eulersche-Φ-Funktion

Beispiel

$\Phi(N) = (p - 1) * (q - 1)$
$\Phi(21) = (3 - 1) * (7 - 1)$
$\Phi(21) = 2 * 6$
$\Phi(21) = 12$

4. Schritt: Verschlüsselungsexponent e

- e teilerfremd und kleiner als $\Phi(N)$
- $1 < e < \Phi(N)$

Beispiel

$e = 5$
$1 < 5 < 12$

5. Schritt: Entschlüsselungsexponent d

- $e * d \equiv 1 \bmod \Phi(N)$
- $(e * d)/\Phi(N)$ selber Rest wie $1/\Phi(N)$

Beispiel

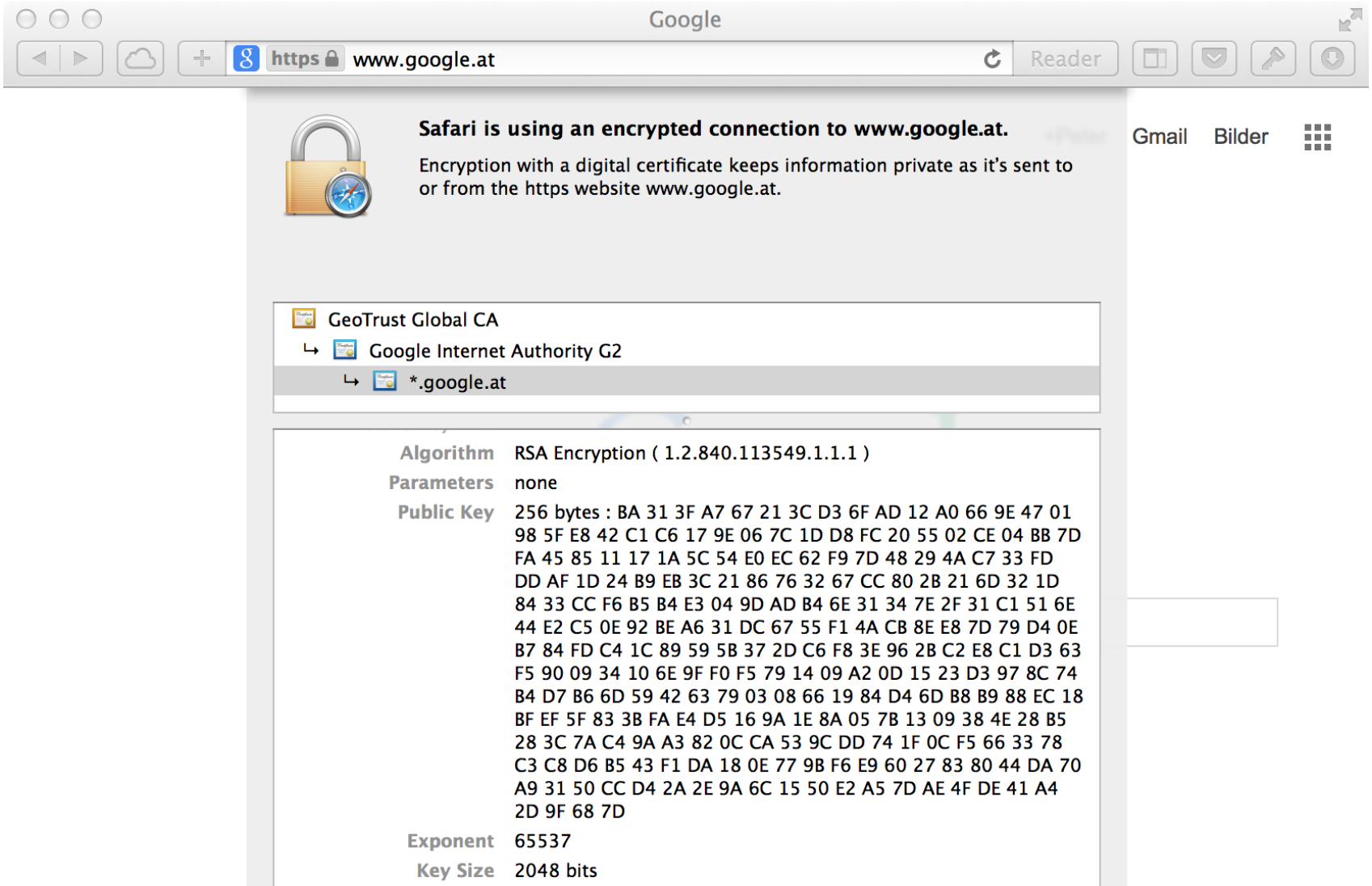
$d = 17$

Privater Schlüssel $\{d, N\} = \{17, 12\}$
Öffentlicher Schlüssel $\{e, N\} = \{5, 12\}$

Schlüssellängen

- Empfehlungen des National Institute of Standards and Technology (NIST)
- bis 2030: 2048 Bit (=256 Byte)
- länger: 3072 Bit (=384 Byte)
- bezieht sich auf RSA-Modulus

Beispiel: Google (2048 Bit)



Beispiel: Google (2048 Bit)

RSA-Modulus:

```
 24 458 322 132 192 074 750 123 737 852 343 849 328 692 208 800 636 145 173 922 806
321 617 698 341 073 783 909 695 464 921 518 119 313 293 932 148 880 300 057 889 205
848 483 755 570 082 737 212 498 434 160 136 633 004 493 539 224 618 110 346 017 649
182 892 756 900 354 388 158 474 708 493 870 410 346 568 279 122 537 117 092 691 941
337 058 147 973 894 824 817 026 340 867 144 834 330 158 411 194 043 194 367 215 194
020 838 742 032 977 944 325 650 945 828 542 512 991 812 085 324 555 728 170 021 446
772 378 824 059 352 307 766 884 155 575 397 046 600 602 949 369 023 746 269 840 657
501 677 422 212 547 004 456 340 395 774 872 667 375 489 925 861 336 516 416 492 486
576 157 016 519 009 389 714 803 133 108 775 909 145 122 246 129 027 471 499 061 271
908 980 435 203 693 686 019 888 056 992 101 018 896 001 264 889 473
```

Beispiel: Google (2048 Bit)

Exponent:
65537

Verschlüsselung

- mit dem öffentlichem Schlüssel $\{e, N\}$
- $c = m^e \bmod N$
- c . . .verschlüsselte Nachricht
- m . . .zu verschlüsselnder Klartext

Verschlüsselung

Beispiel

$c = 4^5 \bmod 21$

$c = 1024 \bmod 21$

$c = 16$

Entschlüsselung

- mit dem privatem Schlüssel $\{d, N\}$
- $m = c^d \bmod N$
- c . . .verschlüsselte Nachricht
- m . . .entschlüsselter Klartext

Entschlüsselung

Beispiel

$m = 16^{17} \bmod 21$

$m = 295147905179352825856 \bmod 21$

$m = 4$

Textnachrichten

Hex	Dec	Char		Hex	Dec	Char	Hex	Dec	Char	Hex	Dec	Char
0x00	0	NULL	null	0x20	32	Space	0x40	64	@	0x60	96	`
0x01	1	SOH	Start of heading	0x21	33	!	0x41	65	A	0x61	97	a
0x02	2	STX	Start of text	0x22	34	"	0x42	66	B	0x62	98	b
0x03	3	ETX	End of text	0x23	35	#	0x43	67	C	0x63	99	c
0x04	4	EOT	End of transmission	0x24	36	$	0x44	68	D	0x64	100	d
0x05	5	ENQ	Enquiry	0x25	37	%	0x45	69	E	0x65	101	e
0x06	6	ACK	Acknowledge	0x26	38	&	0x46	70	F	0x66	102	f
0x07	7	BELL	Bell	0x27	39	'	0x47	71	G	0x67	103	g
0x08	8	BS	Backspace	0x28	40	(	0x48	72	H	0x68	104	h
0x09	9	TAB	Horizontal tab	0x29	41	)	0x49	73	I	0x69	105	i
0x0A	10	LF	New line	0x2A	42	*	0x4A	74	J	0x6A	106	j
0x0B	11	VT	Vertical tab	0x2B	43	+	0x4B	75	K	0x6B	107	k
0x0C	12	FF	Form Feed	0x2C	44	,	0x4C	76	L	0x6C	108	l
0x0D	13	CR	Carriage return	0x2D	45	-	0x4D	77	M	0x6D	109	m
0x0E	14	SO	Shift out	0x2E	46	.	0x4E	78	N	0x6E	110	n
0x0F	15	SI	Shift in	0x2F	47	/	0x4F	79	O	0x6F	111	o
0x10	16	DLE	Data link escape	0x30	48	0	0x50	80	P	0x70	112	p
0x11	17	DC1	Device control 1	0x31	49	1	0x51	81	Q	0x71	113	q
0x12	18	DC2	Device control 2	0x32	50	2	0x52	82	R	0x72	114	r
0x13	19	DC3	Device control 3	0x33	51	3	0x53	83	S	0x73	115	s
0x14	20	DC4	Device control 4	0x34	52	4	0x54	84	T	0x74	116	t
0x15	21	NAK	Negative ack	0x35	53	5	0x55	85	U	0x75	117	u
0x16	22	SYN	Synchronous idle	0x36	54	6	0x56	86	V	0x76	118	v
0x17	23	ETB	End transmission block	0x37	55	7	0x57	87	W	0x77	119	w
0x18	24	CAN	Cancel	0x38	56	8	0x58	88	X	0x78	120	x
0x19	25	EM	End of medium	0x39	57	9	0x59	89	Y	0x79	121	y
0x1A	26	SUB	Substitute	0x3A	58	:	0x5A	90	Z	0x7A	122	z
0x1B	27	FSC	Escape	0x3B	59	;	0x5B	91	[	0x7B	123	{
0x1C	28	FS	File separator	0x3C	60	<	0x5C	92	\	0x7C	124	|
0x1D	29	GS	Group separator	0x3D	61	=	0x5D	93	]	0x7D	125	}
0x1E	30	RS	Record separator	0x3E	62	>	0x5E	94	^	0x7E	126	~
0x1F	31	US	Unit separator	0x3F	63	?	0x5F	95	_	0x7F	127	DEL

Der RSA-Beweis

- erweiterter euklidischer Algorithmus
- kleiner Satz von Fermat
- »Sicherheit« kann nicht bewiesen werden

Erweiterter euklidischer Algorithmus

- euklidischer Algorithmus: größter gemeinsamer Teiler
- Erweiterung gilt wenn $ggT(a, b) = 1$
- Inverse von Kongruenzen lassen sich berechnen $17 * x \equiv 1 \bmod 43$

euklidischer Algorithmus

- kleinere Zahl von größerer subtrahieren

Beispiel

$43 = 17 * 2 + 9$
$17 = 9 * 1 + 8$
$9 = 8 * 1 + 1$

Erweiterung

- von unten aufarbeiten und nach Resten umformen

Beispiel

$$1 = 9 - 8$$
$$8 = 17 - 9$$
$$1 = 9 - (17 - 9)$$
$$1 = 9 - 17 + 9$$
$$1 = 9 * 2 - 17$$

Erweiterung

Beispiel

$$9 = 43 - 17 * 2$$
$$1 = (43 - 17 * 2) * 2 - 17$$
$$1 = 2 * 43 - 4 * 17 - 17$$
$$1 = 2 * 43 - 5 * 17$$

Lösung der Kongruenz

- kürzen (aufgrund der Modulo-Berechnung)
- Addition um positives Ergebnis zu erlangen

Beispiel

$1 = 2 * 43 - 5 * 17$
$1 = -5 * 17$
$\frac{17}{1} \bmod 43 \equiv -5$
$\frac{17}{1} \bmod 43 \equiv 38$

Kleiner Satz von Fermat

- Spezialform des Satzes von Euler
  $a^{\Phi(m)} \equiv 1 \bmod p$
- wenn m Primzahl
- und a kein Vielfaches von m, dann
- $\Phi(p) = p - 1$
- es folgt: $a^{p-1} \equiv 1 \bmod p$

Der Beweis I

- Verschlüsselung: $c = m^e \bmod N$
- Potenzierung von m^e aufheben: $(m^e)^{e^{-1}}$
- Klammern auflösen $m^{e*e^{-1}}$ bzw. $m^{e*\frac{1}{e}}$
- gibt es e^{-1}?

Der Beweis II

- $ggT(a, b) = s * a + t * b$
- $1 = s * \Phi(N) + t * e$
- $t * e \equiv 1 \bmod \Phi(N)$
- t ist gesuchtes e^{-1} und somit bewiesen

Der Beweis III

- gibt es nur ein e^{-1} (Entschlüsselungsexponent d)
- $e * f \equiv 1 \bmod \Phi(N)$
- $e * f * d \equiv d \bmod \Phi(N)$
- es gilt $d * e \equiv 1 \bmod \Phi(N)$
- $f \equiv d \bmod \Phi(N)$

Der Beweis IV

- $1 = s * \Phi(N) + d * e$
- $d * e = 1 - s * (p - 1) * (q - 1)$
- Potenzieren mit Klartext:
  $m^{d*e} = m^{1-s*(p-1)*(q-1)}$
- $m^{s*(p-1)*(q-1)} \equiv 1 \bmod N$
- Satz des Fermat

Sicherheit von RSA

- 3 Teilprobleme dürfen nicht in praktischer Zeit lösbar sein:
- RSA-Problem
- RSA-Schlüsselproblem
- Faktorisierungsproblem

RSA-Problem

- mit öffentlichem Schlüssel $\{e, N\}$ und Geheimtext c
- e-te Wurzel ziehen und modulo N rechnen ergibt Klartext m
- aktuell praktisch nicht lösbar

RSA-Schlüsselproblem

- mit öffentlichem Schlüssel $\{e, N\}$ könnte auf d geschlossen werden
- berechnung der Eulerschen-Φ-Funktion ohne Kenntnis von p und q
- aktuell ebenfalls praktisch unmöglich

Faktorisierungsproblem

- möglicher Angriffsvektor: N in p und q aufteilen (Faktorisierung)
- aktuell nur mit viel Zeitaufwand, aber nicht unmöglich
- Dezember 2009: Faktorisierung von 768 Bit nach 3 Jahren

Padding

- RSA verhält sich deterministisch
- gleicher Output bei gleichem Input
- Nachricht wird um Zufallszahl r erweitert
- Ausgang kann dann nicht mehr analysiert werden

Hybride Kryptographie

- RSA drei Größenordnungen langsamer (10^3) als symmetrische Verfahren
- RSA nur zur Übertragung des Schlüssel der symmetrischen Verfahren
- Sicherheit von RSA, Schnelligkeit von symmetrischen Verfahren
- aber: »Eine Kette ist nur so stark wie ihr schwächstes Glied«

Verwendung von RSA: PGP

- PGP: Pretty Good Privacy
- 1991 von Phil Zimmermann entwickelt
- Verschlüsselung und Signierung von E-Mails

Abbildung : Phil Zimmermann

PGP: Signatur

- kurze Repräsentation (Hash) wird mit privatem Schlüssel verschlüsselt
- jeder kann mit öffentlichem Schlüssel entschlüsseln
- stimmt der Hash überein, kommt E-Mail tatsächlich vom Sender

PGP: Verschlüsselung

- zufälliger symmetrischer Schlüssel für Nachricht
- Schlüssel mit RSA verschlüsselt und angehängt
- somit auch mehrere Empfänger

Die Crypto-Kriege: Anfänge

- 1970er Jahre Kampf um sicherste Verschlüsselungstechniken
- Kalter Krieg: USA gegen die UdSSR
- USA klassifiziert Verschlüsselung als »Waffe«
- somit Exportverbot
- Geheimdienste wollen auch private Daten lesen

Die Crypto-Kriege: 1990er

- US-Kompromiss Clipper-Chip
- Diffie-Hellman Schlüsselaustausch
- Hintertür: Jeder Schlüssel musste bei Behörde hinterlegt werden

Abbildung : Der Clipper-Chip

Die Crypto-Kriege: PGP

- RSA patentiert: Lizenzgebühren
- Exportverbot
- Quellcode als Buch exportiert

Die Crypto-Kriege: Das Ende

- Stärke der Schlüssel limitiert
- Widerstand von Industrie und Banken
- Ende 1990er Jahre Lockerung der Regelungen