# Das Verschlüsselungsverfahren RSA

von Nora Schweppe

Humboldt-Oberschule Berlin

Grundkurs Informatik 3

Herr Dietz

# Inhaltsverzeichnis

# 1. Einleitung

Kryptographie (von dem griechischem Wort *kryptós lógos*, "Verstecktes Wort") ist die Kunst der geheimen Kommunikation. Methoden der Ver– und Entschlüsselung von Nachrichten waren schon vor Jahrtausenden wichtig, als Herrscher nach schnellen und sicheren Nachrichtenwegen suchten, damit dem Gegner geheime Informationen nicht zufielen. Die Evolution der Codes und Chiffren kann als eine Art Wettlauf von rivalisierenden Staaten gesehen werden und hatte somit auch Auswirkungen auf den Gang der Geschichte.
Vor einigen Jahrzehnten wurde Kryptographie fast nur im militärischen Bereich eingesetzt. In der heutigen Informationsgesellschaft jedoch sind Daten und Datenaustausch in allen Bereichen nicht mehr wegzudenken. So wird auch der Schutz von Daten und damit das Wissen über Kryptographie immer wichtiger. Beispielsweise muss beim Home-Banking absolute Sicherheit bestehen, damit nicht Dritte  Zugriff auf Konten anderer erhalten. Heutzutage haben nicht mehr nur Regierungen, sondern auch Großunternehmen und Einzelpersonen ein Interesse an der Verschlüsselung ihrer Daten und Dokumente.

## 1.1 Symmetrische und asymmetrische Verschlüsselungsverfahren

Bei symmetrischen Verschlüsselungsverfahren besitzen Absender und Empfänger einen identischen Schlüssel, der nach außen hin geheim bleibt. Der Absender schickt seine mit dem Schlüssel k in einer Funktion verschlüsselten Nachricht c zum Empfänger. Der Empfänger erhält diesen Geheimtext c und setzt k in die Umkehrfunktion ein, so dass er den Klartext m erhält. Als Beispiel ist die Caesar-Verschlüsselung zu nennen: Der Absender addiert seine Nachricht mit dem Schlüssel. Der Empfänger benutzt die Umkehrfunktion zur Addition, die Subtraktion und erhält so aus dem Geheimtext den Klartext. Ein großes Problem dieses Verfahrens ist der Schlüsselaustausch, der so stattfinden muss, dass es einem Gegner unmöglich ist an den Schlüssel zu gelangen. Die sicherste Methode hierbei ist ein persönliches Treffen, denn Telefone könnten abgehört werden.
Das Problem des Schlüsselaustausches beim symmetrischen Verschlüsseln wurde zur Grundidee der asymmetrischen Verschlüsselung, bei der dieser Austausch nicht mehr nötig ist, da es zwei unterschiedliche Schlüssel gibt. Hierbei hat jede Person einen öffentlichen und einen privaten Schlüssel (Public-Key-Kryptographie). Mit dem öffentlichen Schlüssel wird verschlüsselt und der Besitzer des privaten Schlüssels kann mit diesem entschlüsseln. Dank dem öffentlichen Schlüssel, der für alle Personen sichtbar ist, kann jeder der Person A eine verschlüsselte Nachricht schreiben. Doch nur Person A kann diese Nachricht dank ihrem privaten Schlüssel lesen. Die erste Funktion darf also nicht umkehrbar sein, da sonst jeder mit dem öffentlichen Schlüssel die Funktion umkehren könnte und die geheime Nachricht entschlüs

seln könnte. Auf der anderen Seite muss aber ein Zusammenhang in der Funktion zwischen dem öffentlichen und dem privaten Schlüssel bestehen, da sonst der Empfänger selbst die Nachricht auch nicht entschlüsseln könnte. Trotzdem sollte es unmöglich sein vom öffentlichen auf den privaten Schlüssel zu schließen. Über dieses mathematische Problem wurde lange nachgedacht und bis heute ist es nicht bewiesen, dass es tatsächlich Einwegfunktionen (Funktionen die nicht umkehrbar sind) gibt. In RSA wurde mit einer Trapdoor-Einwegfunktion (trapdoor, *engl.* Geheimtür) gearbeitet, denn dank einer 'Geheimtür' sollte es möglich sein die Funktion doch noch umzukehren.
Diffie und Hellmann waren die ersten denen es 1976 gelang ein Verfahren zum sicheren Schlüsselaustausch zu entwickeln. Entscheidend bei ihrer Idee war die modulo Funktion. Uns allen bekannt ist diese Funktion beispielsweise bei der Uhrzeit. (Es ist 23 Uhr; ich muss in neun Stunden aufstehen; wie spät ist es dann? (23+9)mod 24 → 32 mod 24= 8; um acht Uhr morgens muss ich aufstehen.) Bei ihrem Verfahren gab es eine Unsicherheit, doch ich will hier nicht näher auf dieses Verfahren eingehen. Entscheidend war, dass die Veröffentlichung von Diffie und Hellmann Anregung zur Entwicklung von RSA war.

## 2. Das Verschlüsselungsverfahren RSA

### 2.1 Entstehung von RSA

Ronald **R**ivest, Adi **S**hamir und Leonard **A**dleman, zwei Computerwissenschaftler und ein Mathematiker, befassten sich seit 1976 mit dem Problem der asymmetrischen Verschlüsselung. Seit der Veröffentlichung von Diffie-Hellmann wollten die drei die besagte Trapdoor Einwegfunktion finden. Ist es möglich eine Einwegfunktion zu finden, die nur umgekehrt werden kann, wenn der Empfänger eine besondere Information besitzt? Mit dieser Frage befassten sich vor allem die beiden Computerwissenschaftler, während der Mathematiker Fehler aufspürte und Ideen mathematisch umsetzte. 1977 endlich fand Rivest die Lösung des hier noch einmal kurz erläuterten Problems:
Benennen wir zuvor drei Personen, Alice, Bob und Eve (in der deutschsprachigen Literatur manchmal Erich nach Erich Mielke, Stasi-Chef der DDR), wie sie in der kryptographischen Diskussion genannt werden.

1. Alice erzeugt einen öffentlichen Schlüssel, der für jede Person (auch den Partner Bob und den Feind Eve) sichtbar ist. Dieser Schlüssel muss eine Einwegfunktion sein, es ist also unmöglich für Eve die Funktion umzukehren und die Mitteilungen von Bob an Alice zu entschlüsseln.

2. Alice ist (als einzige) mit ihrem privaten Schlüssel in der Lage den öffentlichen Schlüssel umzukehren um die von Bob zugeschickten Mitteilungen entschlüsseln zu können. Also hat nur Alice die Möglichkeit die an sie gerichteten Nachrichten zu dechiffrieren.

Die Modulo-Funktion spielt eine wichtige Rolle in der asymmetrischen Verschlüsselung von Rivest, da diese Funktion nicht umkehrbar ist. (Berechnet man zum Beispiel 73735 mod 23 = 20, so nützt die Kenntnis der Zahl 20 nicht um die ursprüngliche zahl 73735 zu finden. Es gibt unendlich viele Zahlen, die bei Division durch 23 den Rest 20 ergeben.) Außerdem wird mit der Multiplikation zweier hoher Primzahlen gearbeitet, da angenommen wird, dass Faktorisierung schwierig ist.

RSA ist ein asymmetrisches Verschlüsselungsverfahren in der Form einer Public-Key-Kryptographie (Kryptographie mit einem öffentlichen Schlüssel). Das bedeutet, das ein Schlüssel jedem bekannt sein kann. Entschlüsseln kann die Nachricht aber nur der Besitzer des geheimen privaten Schlüssels.

### 2.2 Das Verfahren und seine Anwendung auf Zahlen

- Man nehme zwei große Primzahlen *p* und *q*.
- Man bilde deren Produkt *n=p*q*, welches Modul genannt wird.
- Man wähle eine Zahl *e* die kleiner als n und teilerfremd (relativ prim) zu *p-1* und *q-1* ist.
- Finde eine Zahl *d*, so dass *(e*d)-1* durch *(p-1)*(q-1)* ohne Rest teilbar ist - durch Umformung erhält man die Bedingung e*d = 1 mod (p-1)(q-1). Die Werte e und d werden öffentlicher und privater Exponent genannt.
- Der öffentliche Schlüssel ist das Paar *(n,e)*, der private Schlüssel ist das Paar *(n,d)*.
- Die Faktoren p und q werden sicher vernichtet oder mit dem privaten Schlüssel zusammen aufbewahrt.

Die Public-Key Eigenschaft, welche besagt dass es unmöglich ist von e auf d zu schließen ist ein Grundkriterium. Außerdem beruht die Sicherheit von RSA auf der Annahme, dass die Faktorisierung von *n* in die Primzahlen *p* und *q* (wenn diese hoch genug gewählt sind) unmöglich ist.

Die Verschlüsselungsfunktion lautet $\boldsymbol{f(m) = c = m^e \bmod n}$, wobei c der Chiffretext und m die Nachricht ist. *mod* ist die Modulofunktion: *x mod y* liefert den Rest bei ganzzahliger Division *x/y*.

Die Entschlüsslung geht prinzipiell ebenso einfach: $\boldsymbol{f^1(c) = c^d \bmod n} = (m^{e*d} \bmod n = m^{(p-1)*(q-1)-1} \bmod n) = \boldsymbol{m}$

Es ist klar, dass man diese Eigenschaft für alle natürlichen Zahlen beweisen muss. Das mache ich hier aber nicht. Es ist der eigentlich schwierige Teil des Verfahrens, der von den Informatikern, die das Verfahren zum ersten Mal vorgestellt haben, bewiesen wurde. (Es beruht auf einem zahlentheoretischen Satz, nämlich der Eulerschen Verallgemeinerung des kleinen Satzes von Fermat.) Im Anhang A findet sich eine Kopie des Beweises aus dem Buch [8.]
Beispiel (hier wähle ich kleine Zahlen, welches beim eigentlichen Verfahren jedoch ungünstig ist, da die Sicherheit auch von der Länge der Primzahlen abhängt):
*p = 5, q = 17. e = 5* ist teilerfremd zu *(p-1) = 4* und *(q-1) = 16, n = p*q = 85*
*(e*d)-1* soll durch *(p-1)*(q-1)* teilbar sein → wählen wir *d = 13 (5*13 –1 / 4*16 = 1)*
Somit ergibt sich der öffentliche Schlüssel *(n,e) = (85,5)* und der private Schlüssel *(n,d) = (85, 13)*
Nun verschlüsseln wir die Zahl 3:

$f(3) = 3^5 \bmod 85 = 243 \bmod 85 = 73 \ (=c)$

Mit dem privaten Schlüssel*(85,13)* kann man nun entschlüsseln:

$f^1(73) = 73^{13} \bmod 85 = 167184950739378888594103 3 \bmod 85 = 3$

Wie man sieht, hat man es sehr schnell mit großen Zahlen zu tun. Ein weiteres Beispiel mit denselben Schlüsseln:

$f(10) = 10^5 \bmod 85 = 100000 \bmod 85 = 40$

$f^1(40) = 40^{13} \bmod 85 = 67108864000000000000 \bmod 85 = 10$

Natürlich ginge das ganze auch umgekehrt: mit dem privaten Schlüssel chiffrieren und mit dem öffentlichen dechiffrieren. Doch das macht keinen Sinn, weil ja jeder den öffentlichen Schlüssel kennt – man könnte also gleich den Klartext verschicken. Doch bei der digitalen Signatur spielt das verschlüsseln mit dem privaten Schlüssel eine Rolle. (In Abschnitt 2.6 werde ich darauf näher eingehen.)
Was i.A. auch nicht möglich ist, den verschlüsselten Text mit dem öffentlichen zu entschlüsseln. Das garantiert das oben benannte Verfahren. Im Beispiel:

$f(40) = 40^5 \bmod 85 = 102400000 \bmod 85 = 75 \ != 10$

Würde man die Funktion so leicht umkehren können wäre das Verfahren völlig unbrauchbar.

Man sieht an den Beispielen schon ein Problem: Der Algorithmus muss mit sehr großen Zahlen rechnen. Ist der private Schlüssel sehr groß, z.B. (7387, 10398), dann muss $c^{7387}$ gerechnet werden. Das führt praktisch immer zum Zahlenüberlauf und damit zum Programmabbruch. Wie man das Problem in den Griff bekommt erkläre ich im fünften Abschnitt dieses Hauptteils. Zunächst soll aber erklärt werden, warum das Verfahren für beliebige Zeichen, nicht nur für Zahlen, gilt.

### 2.3 Anwendung auf Zeichen und Zeichenfolgen

Wie kann man das Verfahren auf andere Daten als Zahlen anwenden? Bekanntlich ist jedes Zeichen in Rechnern binär codiert. Die Buchstaben und andere druckbare Zeichen z.B. als Zahlen zwischen 32 und 255 (ASCII-Codierung). Jede Binärfolge lässt sich aber auch als Zahl interpretieren. Beispiel: Die Zeichenfolge RSA hat den ASCII-Code *828365*, der intern binär dargestellt ist als *0101 0010 0101 0011 0100 0001*

Z.B. ist 82 zur Basis 2 mit acht Stellen statt zur Basis 10 dargestellt: $0*2^7 + 1*2^6 + 0*2^5 + 1*2^4 + 0*2^3 + 0*2^2 + 1*2^1 + 0*2^0 = 64 + 16 + 2 = 82$

Man tut so, als habe man es mit Zahlen zu tun. Die binäre Folge wird einfach als Zahl interpretiert. So kann man einen Text zeichenweise mit RSA verschlüsseln. Auch wenn man keine Chance hat, ein einzelnen Zeichen zu entschlüsseln, ist das trotzdem keine gute Idee: man wendet einfach statistische Entschlüsselungsverfahren an, die wegen der (bekannten) Verteilung von Buchstaben in Texten schnell eine Beziehung zwischen verschlüsselten und unverschlüsselten Zeichen herstellen können. Tritt also ein verschlüsseltes Zeichen etwa so oft auf, wie das Zeichen ’s’ in anderen Texten, dann wurde ziemlich sicher ’s’ verschlüsselt. Stattdessen teilt man den Text in Gruppen zu vier oder mehr Zeichen ein. Es ist dann viel schwerer, mit statistischen Mitteln zu entschlüsseln.

### 2.4 Ist das Verfahren sicher?

Unsicher wäre das Verfahren, wenn man leicht den privaten Schlüssel bestimmen könnte. Das ist möglich, wenn man nur kleine Zahlen verwendet.

Man kennt $n = p*q$ und $e = 5$. Nun überprüft man die (oben genannte) Bedingung $5*e = 1 \mod (p-1)(q-1)$. Da man aber nur $n$ und nicht $p$ und $q$ kennt, muss man zunächst $n$ in zwei Primfaktoren zerlegen. Für kleine Zahlen ist das kein Problem: man testet alle in Frage kommenden Primzahlen, die kleiner oder gleich $n/3$ sind, wenn $3$ kleinstes erlaubtes $p$ ist. Für $n = 85$ testet man nur $3*23$ und $5*17$ und hat schon $p$ und $q$ gefunden. Damit lässt sich der private Schlüssel leicht bestimmen wegen $5*16 \mod 65 = 1$

So einfach das aussieht, so schwierig ist das bei großen Zahlen. Der Grund ist die Schwierigkeit, große Zahlen in Primfaktoren zu zerlegen. Nimmt man eine beliebige Zahl $k$ muss man sie durch alle Primzahlen dividieren, die kleiner sind, als die Quadratwurzel[1] von $k$. Liefert die Division immer einen Rest, ist die untersuchte Zahl eine Primzahl.

[1] Deshalb, weil alle Primfaktoren kleiner als die Quadratwurzel schon getestet wurden . Wenn z.B. k=49 wird geteilt durch 2,3 ,5 (jeweils Rest) und 7 (kein Rest, also 49 keine Primzahl). Dagegen 37: getestet wird 2,3,5 aber sieben schon nicht mehr. Wäre 7 nämlich Primfaktor von 37 müsste 2,3 oder 5 ein anderer sein.

Um eine Zahl mit 1000 Stellen auf die Primzahleigenschaft zu untersuchen, muss man vielleicht durch $10^{100}$ Primzahlen dividieren (nur geschätzt). Nimmt man für jede Division eine Millionstel Sekunden an, braucht man immer noch $10^{100} * 10^{-6} = 10^{94}$ Sekunden das sind , da ein Jahr ungefähr 3600*24*360 = 30 * $10^6$Sekunden hat, mehr als $10^{85}$ Jahre. Selbst die schnellsten Rechner sind nicht in der Lage, die Verschlüsselung zu knacken, sofern nur *p* und *q* groß genug gewählt sind. Da aber die Computer mit der Zeit immer größere Rechenleistungen aufweisen müssen auch die Schlüssel alle paar Jahre vergrößert werden. Bis 1995 wurde angenommen, dass man 50 Jahre benötige um eine Zahl im Bereich von $10^{130}$ in Faktoren zu zerlegen. Doch eine Studie ergab, wenn man einhundert Millionen PCs vernetzen würde (so viele wurden 1995 verkauft), dann könnte eine Zahl in dem Bereich in etwa 15 Sekunden faktorisiert werden. Heute benutzt man bei wichtigen Transaktionen ein n von etwa $10^{308}$, das ist zehn Millionen Milliarden Milliarden Milliarden Milliarden Milliarden Milliarden Milliarden Milliarden Milliarden Milliarden Milliarden Milliarden Milliarden Milliarden Milliarden Milliarden Milliarden Milliarden Milliarden mal größer als $10^{130}$.

Da stellt sich aber die Frage, wie man so große Primzahlen überhaupt finden kann. Das Problem, eine (beliebige) sehr große Primzahl zu finden, ist viel einfacher, als von einer (bestimmten) sehr großen Zahl festzustellen, ob sie prim ist. Dafür gibt es Verfahren in der Mathematik. Diese Eigenschaft ist ganz wichtig für RSA. Wenn jemand ein Verfahren finden könnte, mit der man sehr schnell einen Primzahltest machen könnte, also feststellen, ob eine gegebene große Zahl prim ist, wäre RSA nichts mehr wert, weil man es ganz leicht entschlüsseln könnte.

### 2.5 Potenzieren und Modulo-Berechnung mit großen Zahlen

Selbst bei relativ kleinen Schlüssel wie (5, 493) und (269, 493) muss man $C^{269}$ berechnen. Die kodierte Nachricht c hat dann einen Wert zwischen 0 und 492. Selbst $2^{269}$ auszurechnen, ist auf vielen Rechnern nicht mehr möglich.

Die Berechnung lässt sich aber stark vereinfachen (siehe: [2.])

Es gelten folgende Eigenschaften [8. S. 134]

(I) $(a*b) \bmod c = ((a \bmod c) * (b \bmod c)) \bmod c$

(II) wenn $b = a \bmod c$ dann auch $b^2 = a^2 \bmod c$

Will man z.B. $25^{11}$ mod 23 berechnen, geht man so vor:

**$25^{11}$ mod 23** = ($25^{2\wedge3 +0*2\wedge2 + 2\wedge1+ 1}$) mod 23 = $25^{(((1*2)+0)*2+1)*2)+1}$ mod 23

Nun ist $25^1$ mod 23 = 2

$25^1$ mod 23 = 2 => $25^{1*2}$ mod 23 = 25 mod 23 * 25 mod 23= 4 mod 25 = 4

$25^{1*2+0}$ mod 23 = $((25^1)^2$ mod 23 ) * $25^0$mod 23 = 4* (1 mod 23) = (4*1) mod 23=4

$25^{(1*^{2+0)*2}}$ mod 23 => $25^{(1*^{2+0)*2}}$ mod 23 = $4^2$ (mod 23) = 16

$25^{((1*^{2+0)*2+1)}}$ mod 23 = 16 * ($25^1$ mod 23) = (16* 2 ) mod 23 = 9

$25^{(((1*^{2+0)*2+1)*2}}$ mod 23 => $9^2$ (mod 23 )= 12 mod 23

$25^{(((1*2)+0)*2+1)*2)+1}$ mod 23 = 12* ($25^1$ mod 23 )= (12*2) mod 23 = **1**

Damit wird die modulo-Bildung von großen Potenzen vermieden.

### 2.6 Authentifizieren durch RSA

Ein weiterer Nachteil von symmetrischen Verfahren ist das Problem der Identifikation durch eine digitale Unterschrift. Doch beim asymmetrischen Verfahren RSA ist es dem Absender möglich eine elektronische Signatur unter seine Nachricht zu setzen. Wenn Bob also Alice eine Nachricht schickt signiert er diese digital und damit weiß Alice, dass es nicht die böse Eve war, die die Nachricht verfasst hat um sie auf eine falsche Fährte zu locken. Bob erstellt eine elektronische Unterschrift durch Potenzieren:

$s = m^d \ mod \ n$ , wobei *(n,d)* Bobs privater Schlüssel ist und *s* seine Signatur

Nach Erhalten der Nachricht mit Signatur prüft Alice die Unterschrift mit Bobs öffentlichem Schlüssel *(n,e)* nach:

$s^e \ mod \ n = m$

Jede Person hat mit dem öffentlichen Schlüssel des Absenders die Möglichkeit die Signatur auf ihre Echtheit zu überprüfen, doch nur der Absender kann mit seiner Signatur unterschreiben, denn kein anderer besitzt seinen privaten Schlüssel.

Trotzdem muss zuvor die geheime Nachricht mit dem öffentlichen Schlüssel des Empfängers chiffriert werden (wie bereits oben erläutert). Nur die digitale Signatur kann mit dem eigenen privaten Schlüssel stattfinden.

## 3. Resultierende Effizienz von RSA

RSA ist weitaus sicherer und bequemer als symmetrische Verfahren, bei denen ein Schlüsselaustausch nötig ist. Durch die elektronische Unterschrift kann die Authentizität einer Nachricht gegeben werden, welches in symmetrischen Chiffrierverfahren sehr schwierig und nur durch das Vertrauen einer dritten Instanz möglich ist. Außerdem hat bei RSA jeder Nutzer die eigene Verantwortung über seinen geheimen Schlüssel und durch sorgfältiges Aufbewahren kann dieser nie in die Hände von anderen Personen fallen. So gut dies klingt, so groß ist dennoch das Geschwindigkeitsproblem, da, wie oben gezeigt, mit immens großen Zahlen gerechnet wird.

Die asymmetrische Kryptographie will die symmetrische also nicht ablösen, sondern unterstützen und sicherer machen. Durch eine Kombination beider Systeme kann ein sogenannter *digitaler Umschlag* erstellt werden. Der Schlüssel eines symmetrischen Verfahrens wird hierbei asymmetrisch verschlüsselt und zum Empfänger gesandt. Da der Schlüssel im geringsten nicht so groß sein wird wie die zu sendende Nachricht ergibt sich ein Geschwindigkeitsvorteil. Durch die asymmetrische Verschlüsselung ergibt sich die Sicherheit, dass den Schlüssel niemand ungewolltes empfangen kann. Mit dem Schlüssel kann der Empfänger nun die symmetrisch verschlüsselte Nachricht entschlüsseln.
RSA ist dementsprechend sehr effizient für z.B. Regierungen und Großunternehmen, denen eine absolute Sicherheit am wichtigsten ist. Doch für Einzelpersonen hat RSA nicht die gewünschte Effizienz, da es für sie zu langsam sein wird bei einer nicht allzu wichtigen Nachricht.

# Literaturverzeichnis


**1.** Beutelsbacher, Albrecht; Schwenk, Jörg; Wolfenstetter, Klaus-Dieter: Moderne Verfahren Kryptographie. Von RSA zu Zero-Knowledge. 4. Auflage *Vieweg Verlagsgesellschaft* 2001

**2.** Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.: Introduction To Algorithms. *The MIT Press* Cambridge London 1990

**3.** Ertel, Wolfgang: Der RSA Algorithmus. *http://www.sias.de/kryptologie-rsa.html.* 2001

**4.** RSA by RSA Security. *http://www.rsasecurity.com/rsalabs/faq/3-1.html.* RSA Security Inc. 2002

**5.** Singh, Simon: Geheime Botschaften. *Carl Hanser Verlag Münschen* Wien 2000

**6.** Spolwig, Sigfried: Kryptographie. *http://www.oszhdl.be.schule.de/gymnasium/faecher/informatik/krypto/.* April 2002

**7.** Thöing, Christian: Das Verfahren RSA. *http://mitglied.lycos.de/cthoeing/crypto/rsa.htm.* 2002

**8.** Van der Lubbe, Jan: Basic Methods Of Cryptography. *Cambridge University Press* 1998, S. 131 - 145

# Anhang A

aus [8.] S. 134 – 135

Eulerschen Verallgemeinerung des kleinen Satzes von Fermat

**Theorem 6.1 (Euler's theorem)**
For all $a$ and $n$, which are relatively prime with respect to each other and for which $n > 0$ and $0 < a < n$, it holds that:

$$a^{\varphi(n)} = 1 \pmod{n}. \tag{6.8}$$

**Proof**
For a given $n$, $a \in Z'_n$, with $Z'_n = (r_1,\ldots,r_m)$ representing the set of all positive integers between 0 and $n$ which are relatively prime to $n$. It is evident that by definition:

$$|Z'_n| = m = \varphi(n).$$

For all values of $i$, there must be a value of $j$ $(1 \le i, j \le m)$ such that

$$ar_i = r_j \pmod{n}.$$

This is explained as follows. Since $a$ and $r_i$ are relatively prime to $n$, the product $ar_i$ must also be relatively prime to $n$. The set $Z'_n$ includes all the numbers which are relatively prime to $n$, so therefore $ar_i \pmod{n} \in Z'_n$. In other words, $ar_i \pmod{n}$ is identical to one of the elements $r_j$ of $Z'_n$. For every $i$, there is exactly one $j$ such that $ar_i = r_j \pmod{n}$.
It follows that

$$ar_1 ar_2 \ldots ar_m = r_1 r_2 \ldots r_m \pmod{n}$$

or

$$a^m(r_1 r_2 \ldots r_m) = (r_1 r_2 \ldots r_m) \pmod{n},$$

resulting in

$$(a^m - 1)(r_1 r_2 \ldots r_m) = 0 \pmod{n}.$$

Since $(r_1 r_2 \ldots r_m)$ are relatively prime to $n$, we find that

$$a^m - 1 = 0 \pmod{n} \Rightarrow a^m = 1 \pmod{n}.$$

Theorem 6.1 follows immediately by substituting $m = \varphi(n)$ in this last statement. □

**Example**
Let $n = 17$ and $a = 2$. Then $\varphi(17) = 16$. Euler's theorem states that $2^{16} = 1 \pmod{17}$. Straightforward calculation confirms that this is correct, $2^{16} = 65536 = 1 + 3855 \times 17$. △

Euler's theorem can be used to prove that encipherment and decipherment using the RSA system are the inverse operation of each other.