# SOURCE: http://www.di-mgt.com.au/rsa_factorize_n.html

# RSA: how to factorize N given d

---

This page explains how to factorize the RSA modulus $N$ given the public and private exponents, $e$ and $d$.

## Introduction

For the RSA algorithm, we have a public key $(N, e)$ and a private key $(N, d)$ where $N = pq$ is the product of two distinct primes $p$ and $q$, and the numbers $e$ and $d$ satisfy the relation $ed \equiv 1 \mod \phi(N)$ where $\phi(N) = (p-1)(q-1)$. $N$ should be a large number which is impossible to factorize, typically of length 1024 bits. The numbers $N$ and $e$ can be made public, but $d$, $p$, $q$ and $\phi(N)$ are kept secret by the user of the private key. See our RSA Algorithm and RSA Theory pages for more information.

## The problem: given *d* and *e*, can we factorize *N*?

Surprisingly, there isn't a simple formula to compute the factors $p$ and $q$ of the modulus $N$ given just the public and private exponents, $e$ and $d$. But there is a nice efficient algorithm using a random $g$ which should succeed about half the time.

Initially we compute $k = de - 1$. We then choose a random integer $g$ in the range $1 < g < N$. Now $k$ is an even number, where $k = 2^t r$ with $r$ odd and $t \geq 1$, so we can compute $x = g^{k/2}, g^{k/4}, \ldots, g^{k/2^t} \pmod N$ until $x > 1$ and $y = \gcd(x-1, N) > 1$. If so, then one of our factors, say $p$, is equal to $y$, and the other is $q = N/y$ and we are done. If we don't find a solution, then we choose another random $g$.

---

### Algorithm

---

**Input:** $N, e, d$.
**Output:** $p$ and $q$ where $pq = N$.

1. [Initialize] Set $k \leftarrow de - 1$.
2. [Try a random g] Choose $g$ at random from $\{2, \ldots, N-1\}$ and set $t \leftarrow k$.
3. [Next t] If $t$ is divisible by 2, set $t \leftarrow t/2$ and $x \leftarrow g^t \mod N$. Otherwise go to step 2.
4. [Finished?] If $x > 1$ and $y = \gcd(x-1, N) > 1$ then set $p \leftarrow y$ and $q \leftarrow N/y$, output $(p, q)$ and terminate the algorithm. Otherwise go to step 3.

---

## A simple example

**Input:** $N = 25777$, $e = 3$, $d = 16971$.

```
k=de-1=50912
```

```
Trying g=2
```

```
t=25456
x=g^t mod N=1
t=12728
x=g^t mod N=1
t=6364
x=g^t mod N=1
t=3182
x=g^t mod N=25776
y=gcd(x-1,N)=1
t=1591
x=g^t mod N=12709
y=gcd(x-1,N)=1

Trying g=5
t=25456
x=g^t mod N=1
t=12728
x=g^t mod N=1
t=6364
x=g^t mod N=1
t=3182
x=g^t mod N=15050
y=gcd(x-1,N)=149
p=149
q=N/p=25777/149=173

Output: p=173, q=149
```

Note that we swapped $p$ and $q$ here in accordance with the convention that $p > q$.

# Code to do this with large integers

We use our BigDigits multiple-precision arithmetic software to implement this algorithm for large integers. The code is here. Note that we cheat slightly by just choosing small primes $g = 2, 3, 5, 7, 11, \ldots$ instead of random values for $g$. We should get a result within a few tries.

The output for the 508-bit example from [KALI93] should be as follows:

```
Input:
n=a66791dc6988168de7ab77419bb7fb0c001c62710270075142942e19a8d8c51d053b3e3782a1de
5dc5af4ebe99468170114a1dfe67cdc9a9af55d655620bbab
e=10001
d=123c5b61ba36edb1d3679904199a89ea80c09b9122e1400c09adcf7784676d01d23356a7d44d6b
d8bd50e94bfc723fa87d8862b75177691c11d757692df8881
k=de-1=39120867845114712895611519523019250861689464547093160959606188007427993691
73365724802767882894790378640224771497700744632822349777243736269522672978216650880
0880
Trying g=2
k1=1956043392255735644780575976150962543084473227354658047980309400371399684586
6828624013839414473951893201123857488503723164111748886218681347613364891083254 40
x=g^{k1} mod N=1
k1=9780216961278678223902879880754812715422366136773290239901547001856998422933 4
1431200691970723697594660056192874425186158205587444310934067380668244554162720
x=g^{k1} mod N=1
k1=4890108480639339111951439940377406357711183068386645119950773500928499211466 7
0715600345985361848797330028096437212593079102793722155467033690334122277081360
x=g^{k1} mod N=1
k1=2445054240319669555975719970188703178855591534193322559975386750464249605733 3
5357800172992680924398665014048218606296539551396861077733516845167061138540680
x=g^{k1} mod N=1509343371826844042603130722622996730453128009662463183280213346 7
7550393256965570068078640428100663900924739937728765865221716187647865923600986 3
49707225
y=gcd(x-1,N)=2323495016218899338815592763008533131685106005533447038236880433183
4850828939
Output:
```

```
p=33d48445c859e52340de704bcdda065fbb4058d740bd1d67d29e9c146c11cf61
q=335e8408866b0fd38dc7002d3f972c67389a65d5d8306566d5c4f2a5aa52628b
```

which is indeed the correct factorization.

## References


- **[BONE99]** Boneh, D. *Twenty Years of Attacks on the RSA Cryptosystem*, Notices of the American Mathematical Society, 46(2):203-213, 1999, <link>
- **[KALI93]** Burton Kalinski. *Some Examples of the PKCS Standards*, RSA Laboratories, 1999, <link>.


## Contact us

To comment on this page or to tell us about a mistake, please send us a message.

*This page first published 1 December 2012. Last updated 24 December 2012.*

---