Bob thinks that $p$ and $q$ are primes but $p$ isn’t. Then, Bob thinks $\Phi_{\text{Bob}} := (p-1)(q-1) \neq \phi(n)$. Is this true?

Bob chooses a random $e$ ($1 < e < \Phi_{\text{Bob}}$) such that $\gcd(e, \Phi_{\text{Bob}}) = 1$. Then, $d = e^{-1} \mod \Phi_{\text{Bob}}$.

Example:
$p = 9, q = 5, \Phi_{\text{Bob}} = 32, e = 3$, then $d = 11$.

What is the encryption of $2$?
$m^{2} = 2^{3} \equiv 8 \mod 45$
ciphertext: $8$

What is the decryption of the encryption of $2$?
$8^{d} \mod 45 = 17$
not the plaintext!
RSA Question 3

Let’s compute a table of the products of all \( \leq 512 \)-bit primes.

If we have such a table, how do we find our \( n \)?

But how much space would such a table need?

- How many primes of \( \leq 512 \) bits?

- How many entries in our table?

- How many bits per entry?

- Total number of bits?

Atoms in the universe: about \( 10^{80} \)
Breaking RSA

Obvious ways how to try to break RSA:

- Factor $n$.
- Compute $\Phi(n)$.
- Compute $d$.

Observation: Computing $\Phi(n)$ is not easier than factoring $n$.

\[ n = 77 \]
\[ \Phi(n) = 60 \]
\[ n = p \cdot q \]
\[ \Phi(n) = (p-1)(q-1) \]
\[ n - \Phi(n) = p + q - 1 \]
\[ p + q - 1 = 77 - 60 \]
\[ p \cdot q = 18 \]
\[ p = 18 - q \]
\[ p \cdot q = (18 - q) \cdot q = n = 77 \]
Breaking RSA

Obvious ways how to try to break RSA:

- Factor n.
- Compute $\Phi(n)$.
- Compute d.

Note: If we know d, then we can use a polynomial-time randomized algorithm to factor n (we will not do this). Thus, computing d is not easier than factoring n.

Thus: We will try the factoring approach.

Note: Does this mean that breaking RSA is as hard as factoring? Not necessarily (remote possibility of decrypting via some new magical formula without finding d or p, q?).
Factoring Algorithms

- many algorithms (no polynomial-time!)
- the most effective on very large numbers (and their running times):

  - quadratic sieve: $O(e^{(1+o(1))(\ln n \cdot n \ln n)^{1/2}})$

  - elliptic curve: $O(e^{(1+o(1))(2 \ln p \cdot \ln \ln p)^{1/2}})$

  - number field sieve: $O(e^{(1.92+o(1))(\ln n)^{1/3} (\ln \ln n)^{2/3}})$

where $p$ is denotes the smallest prime factor of $n$

Note: For RSA, quadratic sieve is better than elliptic curve, number field sieve is even faster (for very large numbers). In 1999 it was used to factor RSA-155 (a 155-bit number).
Simple Factoring Algorithms

We will look at some simpler factoring algorithms. We will usually find one non-trivial factor of n – how to get the complete factorization?

The simplest factoring algorithm: trail division.

\[ \text{for } i = 2 \text{ to } \sqrt{n} : \]
\[ \quad \text{if } i \text{ divides } n, \text{ return } i \]
\[ \quad \text{return prime} \]

Running time: \( O(\sqrt{n}) \), not a polynomial time

Does it work?

Yes, but...
Simple Factoring Algorithms

**Pollard p-1 factoring algorithm** \((n,B):\)
Input: odd number \(n\), bound \(B\)

1. \(b := 2\)
2. for \(j\) from 2 to \(B\) do:
3. \(b := b \cdot j \mod n\)
4. \(d := \gcd(b-1,n)\)
5. if \(1 < d < n\) then:
6. return \(d\)
7. return “failure”

Note: From 1974, see page 182. Sometimes works for larger integers.
Simple Factoring Algorithms

What is the running time of Pollard p-1?

\[ O( B \cdot \log n \cdot \log \log B) + \log n ) \]

If \( B \) is reasonably small \( \approx \log \log n \) OK

\( B = n \) too \( \approx \log n \) (exponential)

\textbf{Yes}

Note:
Works only if \( n \) has a prime factor \( p \) such that \( p - 1 \) has only "small" prime factors.
For example for RSA, we can have \( p, q \) such that \( p = 2p_1 + 1 \) and \( q = 2q_1 + 1 \) where \( p_1 \) and \( q_1 \) are also primes.

Note:
Lenstra’s elliptic curve method is a generalization of this algorithm (we will not go into it).
Another Factoring Algorithm

Idea (base for quadratic sieve and number field sieve):
If \( x \neq \pm y \pmod{n} \) and \( x^2 \equiv y^2 \pmod{n} \),
then \( \gcd(x+y,n) \) and \( \gcd(x-y,n) \) are nontrivial factors of \( n \).

Example:

\[
10^2 \equiv 32^2 \pmod{77}
\]

So: \( \gcd(10+32,77) = 7 \) and \( \gcd(10-32,77) = 11 \) are nontrivial factors of 77

The catch: how to find \( x \) and \( y \) ?
Another Factoring Algorithm

Many algorithms use a factor base: a set of the smallest b primes.

\[ b = 5 \text{ factor base: } 2, 3, 5, 7, 11 \]

Idea:
- Suppose we find several numbers \( z \) such that all prime factors of \( z^2 \mod n \) are in the factor base.
- Take a product of several of these \( z^2 \)'s so that each prime in the factor base is used an even number of times.
- That will give us \( x^2 \equiv y^2 \pmod{n} \).
- If \( x \not\equiv \pm y \pmod{n} \), we get a nontrivial factor of \( n \).
Another Factoring Algorithm

Example 1:
Let \( n = 197209 \) and let \( \{2, 3, 5\} \) be our factor base. Consider the following \( z \)'s: \( 159316 \) and \( 133218 \).

\[
\begin{align*}
159316^2 &\equiv 2^4 \cdot 3^2 \cdot 5^1 \pmod{n} \\
133218^2 &\equiv 2^0 \cdot 3^4 \cdot 5^1 \pmod{n}
\end{align*}
\]

Then,
\[
\begin{align*}
(159316 \cdot 133218)^2 &\equiv (2^2 \cdot 3^3 \cdot 5)^2 \pmod{n}.
\end{align*}
\]

Reducing both sides mod \( n \), we get \( 126308^2 \equiv 540^2 \pmod{n} \).

Use the Euclidean algorithm to find \( \gcd(126308-540,n)=199 \).

We obtain the factorization \( 197209 = 199 \cdot 991 \).
Another Factoring Algorithm

Example 2:
Let \( n = 15770708441 \) and let \( \{2, 3, 5, 7, 11, 13\} \) be our factor base. Consider the following z’s: 8340934156, 12044942944, and 2773700011.

\[
\begin{align*}
8340934156^2 &\equiv 3 \cdot 7 \pmod{n} \\
12044942944^2 &\equiv 2 \cdot 7 \cdot 13 \pmod{n} \\
2773700011^2 &\equiv 2 \cdot 3 \cdot 13 \pmod{n}
\end{align*}
\]

Then,

\[
(8340934156 \cdot 12044942944 \cdot 2773700011)^2 \equiv (2 \cdot 3 \cdot 7 \cdot 13)^2 \pmod{n}.
\]

Reducing mod n, we get \( 9503435785^2 \equiv 546^2 \pmod{n} \)

Then, \( \gcd(9503435785-546,n) = 115759 \).
Another Factoring Algorithm - Vectors

Suppose the factor base is \{p_1, \ldots, p_b\} and let \(z\) be such that all prime factors of \(z^2 \mod n\) are in the factor base, e.g.,

\[ z^2 \mod n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_b^{\alpha_b} \]

How many different \(z\)'s do we need?

For each \(z\), look at the binary vector \((\alpha_1 \mod 2, \ldots, \alpha_b \mod 2)\).

For example, in Example 1, the binary vector for 159316 is \((4 \mod 2, 2 \mod 2, 1 \mod 2) = (0,0,1)\) and the vector for 133218 is \((0 \mod 2, 4 \mod 2, 1 \mod 2) = (0,0,1)\).

We need a set of \(z\)'s that the sum of their binary vectors is \((0,\ldots,0)\). In Example 1: \((0,0,1) + (0,0,1) \equiv (0,0,0) \pmod{2}\).

What are the vectors for Example 2?
Another Factoring Algorithm - How many z's

Let b be the size of the factor base. Suppose we have c z's. We need to have a set of vectors (one vector per z) so that they sum to (0,...,0). How large does c need to be?

Let's consider an example:

\[ v_8 \equiv v_2 + v_5 + v_6 + v_7 \pmod{2} \]

\[ \Rightarrow v_2 + v_5 + v_6 + v_7 + v_8 \equiv (0,0,...,0) \]

We can have up to 7 (b) linearly independent vectors.

If \( c > b \), then we can describe one of the vectors as a linear combination of the other vectors:

\[ v_8 \equiv \beta_1 v_1 + \beta_2 v_2 + \beta_3 v_3 + \ldots + \beta_7 v_7 \pmod{2} \]

for all \( \beta_i \in \{0,1\} \)

Well... we also need to worry about the condition \( x \neq \pm y \pmod{n} \) - recall that we already have \( x^2 \equiv y^2 \pmod{n} \). The good news is that \( x \neq \pm y \pmod{n} \) happens with probability \( \leq \frac{1}{2} \). (I.e. take a little larger c and it's ok.)
Another Factoring Algorithm – Choosing z’s

There are different ways to choose the z’s.

In the random squares algorithm, the z’s are chosen randomly.

Another possibility is to try integers of the form $j + \lfloor (in)^{1/2} \rfloor$ for small $j$ and $i=1,2,3,...$
These integers tend to be small when squared mod $n$.

Note: quadratic sieve uses a sieving procedure to determine the z’s.

Choosing the factor base: all primes smaller than $2^{(\log n \log \log n)^{1/2}}$. 