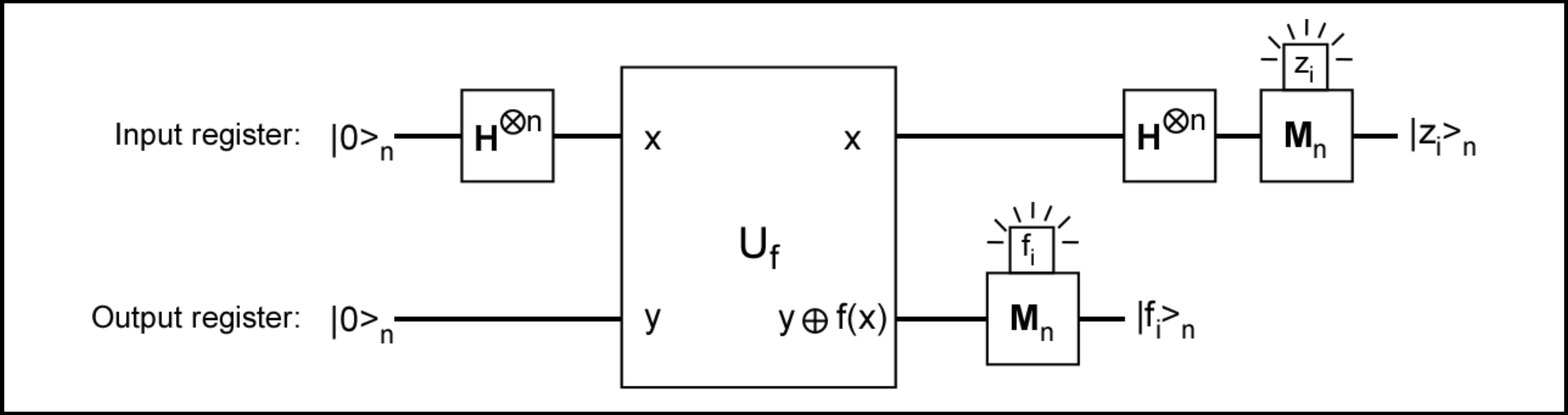
RSA Encryption, Factoring, and Period Finding

Background: In 1993 Dan Simon found a quantum algorithm that can efficiently find a hidden "period" *a* for a function defined by

$$f(x \oplus a) = f(x)$$



Shortly afterwards, in 1994, Peter Shor used a very similar approach to finding the hidden period of a function

$$f(x + k) = f(x)$$

Goal: Explain why that is an interesting problem!

A. Why is factoring and period finding interesting?

There is a relationship between finding the period of a function

$$f(x) = a^x \pmod{\mathrm{N}}$$

$$a^{x+k} \equiv a^x \pmod{\mathrm{N}}$$

For an integer N = pq, where p and q are primes and finding the prime factors p and q of N.

And why is *that* interesting?

Public Key encryption relies on the fact that prime factorization of a composite number is a “hard” (exponential in the number of bits) problem.

1. Encryption and its importance

a. Private Key Encryption

In private key encryption two parties (Alice and Bob, of course) know a “secret key” that allows them to both encrypt and decrypt a message.

A perfectly secure system is a “one-time pad” containing a list of random characters, one for each character in the message.

Example: Consider the letter “M” (message) in ASCII encoding is:

$$\text{``}M\text{''} = 0100\ 1101$$

And the secret code character “C” which in ASCII is:

$$\text{``}C\text{''} = 0100\ 0011$$

The encoded message is the XOR of M and C.

$$E = M \oplus C = 0000\ 1110$$

The decoded message is the XOR of E and C.

$$D = E \oplus C = 0100\ 1101 = \text{``}M\text{''}$$

b. Public Key Encryption

In public key encryption there are two keys, a public key used for encoding and a private key used for decoding.

The key (sorry) to public key encryption is that even if you know the public key it is hard (mathematically hard) to figure out what the private key is, but if you generate a private key it is easy (mathematically easy) to generate the associated public key.

Every time you do an online transaction – sending your credit card number to a company via your web browser you use public key encryption. In particular you use RSA (named RSA after its inventors Rivest, Shamir, and Adleman who designed it in 1977).

In RSA the private (decoding) key requires knowledge of two prime numbers p and q and the public (encoding) key requires knowledge of only the composite number $N = pq$. The fact that *finding primes* is easy, *creating a composite* is easy, and *factoring a composite* is hard is the basis for RSA encryption.

B. The mathematics of groups (as related to RSA)

1. Modular division

Definition:

$$a \text{ (mod N)} = a - \left[\frac{a}{N}\right] N$$

Integer Division

a (mod N) is the remainder when a is divided by N

Examples:

$$323 \text{ mod } 21 = 323 - 15 * 21 = 8$$
$$646 \text{ mod } 21 = 646 - 30 * 21 = 16$$

Properties:

$$(a + b) \text{ (mod N)} = (a \text{ (mod N)}) + (b \text{ (mod N)}) \text{ (mod N)}$$

$$ab \text{ (mod N)} = (a \text{ (mod N)})\,(b \text{ (mod N)}) \text{ (mod N)}$$

Example 1:

$$(323 + 646) \text{ (mod 21)} = (8 + 16) \text{ (mod 21)}$$
$$= 24 \text{ (mod 21)} = 3$$

Example 2:

$$323 \cdot 5 \text{ (mod 21)} = (8)(5) \text{ (mod 21)}$$
$$= (40) \text{ (mod 21)} = 19$$

a. Calculating the GCD using Euclid’s Algorithm

If you want to calculate the greatest common denominator of *a* and *b*

Iterate the relations

$$a_{n+1} = b_n \qquad b_{n+1} = a_n \ (\mathrm{mod}\ \mathrm{b}_n)$$

from n = 0 until b_n = 0. The GCD is the final value of a_n.

Example: What is the GCD of 2947 and 77?

$$\begin{array}{lll} n = 0 & a_0 = 2947 & b_0 = 77 \\ n = 1 & a_1 = 77 & b_1 = 21 \\ n = 2 & a_2 = 21 & b_2 = 14 \\ n = 3 & a_3 = 14 & b_3 = 7 \\ n = 4 & a_4 = 7 & b_4 = 0 \end{array}$$

GCD (2947, 77) = 7.

An Excel version of this iterative routine is available on Moodle

2. Groups

A group G_N is a set of numbers $\{c_i\}$ that

1. Contains the number 1
2. Is closed under multiplication mod N. That is, for all elements c_i and c_j their product is also a member of the group c_k.

$$c_i c_j \equiv c_k \text{ (mod N)}$$

3. Each element c_i has a multiplicative inverse d_i

$$c_i d_i \equiv 1 \text{ (mod N)}$$

that is also part of the group.

a. A group G_N contains all of the integers less than it that are relatively prime to it; the gcd of each element c and N is 1.
b. The number of elements in a group is called the ORDER of the group.

Example:

$$G_{21} = \{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$$

$\longleftarrow$ $O_{21} = 12.$

$$17 \cdot 19 \text{ (mod 21)} = 323 \text{ (mod 21)} = 8$$

$$17 \cdot 5 \text{ (mod 21)} = 85 \text{ (mod 21)} = 1$$

a. Aside on calculating the modular inverse:

Given an element *c* of a group G_N , what value *d* gives

$$cd \equiv 1 \text{ (mod N)}$$

This can be found with the Extended Euclid's Algorithm

If you want to calculate the inverse of c (mod N)

Start with the relations

$$N_0 = N \qquad c_0 = c \qquad q_0 = N_0/c_0 \qquad d_0 = 0$$
$$N_1 = c_0 \qquad c_1 = N_0 \text{ (mod } c_0) \qquad q_1 = N_1/c_1 \qquad d_1 = 1$$

Then iterate:

$$N_{n+1} = c_n \qquad c_{n+1} = N_n \text{ (mod } c_n)$$
$$q_{n+1} = N_{n+1}/c_{n+1} \qquad d_{n+1} = d_{n-1} - d_n q_{n-1} \text{ (mod N)}$$

from n = 3 until $c_n = 0$. Note that N, c, and q are the same as in Euclid's algorithm; the inverse modulo N is the final value of d_n.

An Excel version of this iterative routine is available on Moodle.
An alternative approach is described in Appendix J. 2 of Mermin.

Example, what is the modular inverse of c = 8 (mod 21):

n	N_n	c_n	q_n	d_n
0	21	8	2	0
1	8	5	1	1
2	5	3	1	19
3	3	2	1	3
4	2	1	2	16
5	1	0		8

$-2 \pmod{21}$

$= 21 - 2 = 19$

$d_0 = 0$

$d_1 = 1$

$d_2 = d_0 - d_1 q_0 \pmod{21}$

$d_3 = d_1 - d_2 q_1 \pmod{21}$

$\cdots$

$\longleftarrow$ d is the value of d_n when $c_n = 0$.

Answer: d = 8 (mod 21)

Check: $8 \cdot 8 \pmod{21} = 64 \pmod{21} = 1$

b. A Group G_p with N a prime number *p* contains all of the integers less than *p*

$$G_p = \{1, 2, 3, ..., p-2, p-1\}$$

For G_p the number of elements is p-1.

Fermat's Little Theorem: For every element a in G_p

$$a^{p-1} \equiv 1 \text{ (mod p)}$$

A group $G_{N=pq}$ with N a product of two prime numbers p and q has the order (p-1)(q-1).

Example:

$$G_{21} = G_{7 \cdot 3}$$

The order of G_{21} is (7-1)(3-1) = 12.

Extension to Fermat's Little Theorem: For every element *a* in G_{pq}

$$a^{(p-1)(q-1)} \equiv 1 \text{ (mod pq)}$$

Example:

$$4^{12} \equiv 1 \text{ (mod 21)}$$

c. Aside on calculating numbers like

$$19^{12} \pmod{21} = 1$$

using "repeated squaring."

Note that 12 = 1100 in binary, so

$$19^{12} = 19^{8+4} = (19^8)^1 \cdot (19^4)^1 \cdot (19^2)^0 \cdot (19^1)^0.$$

Repeated Squaring

$$19^1 \pmod{21} = 19$$

$$19^2 \pmod{21} \equiv 361 \pmod{21} \equiv 4 \pmod{21}$$

$$19^4 \pmod{21} \equiv 4^2 \pmod{21} \equiv 16 \pmod{21}$$

$$19^8 \pmod{21} \equiv 16^2 \pmod{21} \equiv 256 \pmod{21} = 4$$

$$\begin{aligned} 19^{12} \pmod{21} &\equiv 4^1 \cdot 16^1 \cdot 4^0 \cdot 19^0 \pmod{21} \\ &\equiv 256 \pmod{21} \equiv 1 \end{aligned}$$

An Excel version of this iterative routine is available on Moodle

d. Subgroups

A subgroup of G_N is a subset of group elements that ALSO satisfies the rules of the group.

Example #1:

$$S_{21}^{(4)} = \{1, 4, 16\}$$

$$4 \cdot 4 \pmod{21} = 16 \qquad 16 \cdot 4 \pmod{21} \equiv 64 \pmod{21} \equiv 1$$

Example #2:

$$S_{21}^{(8)} = \{1, 8\}$$

Example #3:

$$S_{21}^{(11)} = \{1, 2, 4, 8, 11, 16\}$$

Lagrange’s Theorem:

The order of any subgroup is a divisor of the order of the group.

In the examples, the order of the first subgroup is 3 = 12/4,
the second is 2 = 12/6 and the third is 6 = 12/2.

e. Generating sub-groups by powers of the member of a group.

Each member *a* of the group can create a sequence of numbers

$$a^n \pmod{\mathrm{N}} \qquad n = 0, 1, 2, 3, \ldots$$

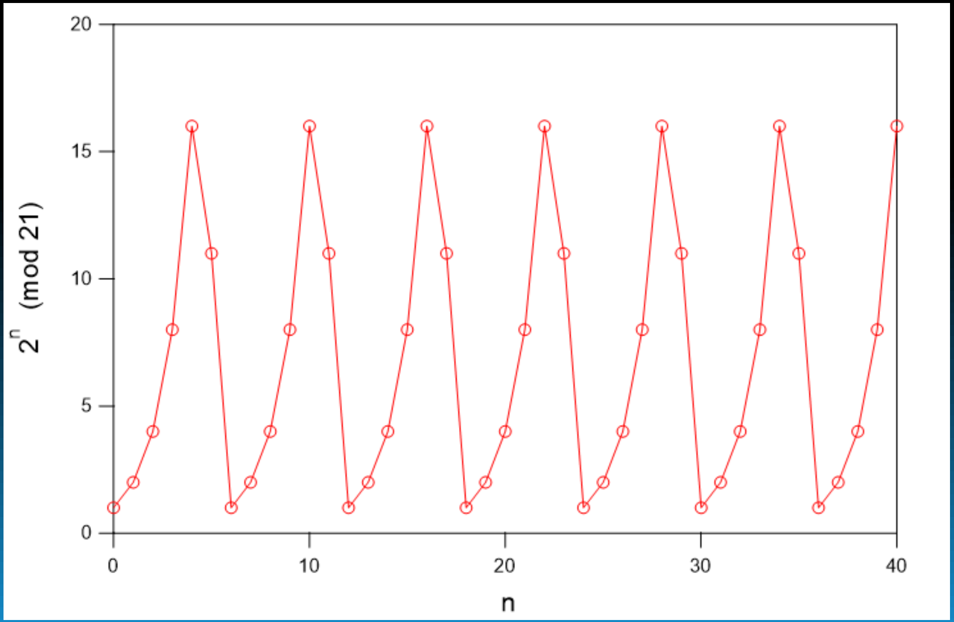
Which will always generate members of the group, and repeats after a period (called the order of the group member) *k*.

Example:

$$2^0 \bmod 21 = 1$$
$$2^1 \bmod 21 = 2$$
$$2^2 \bmod 21 = 4$$
$$2^3 \bmod 21 = 8$$
$$2^4 \bmod 21 = 16$$
$$2^5 \bmod 21 = 11$$
$$2^6 \bmod 21 = 1$$
$$2^7 \bmod 21 = 2$$
$$\ldots$$

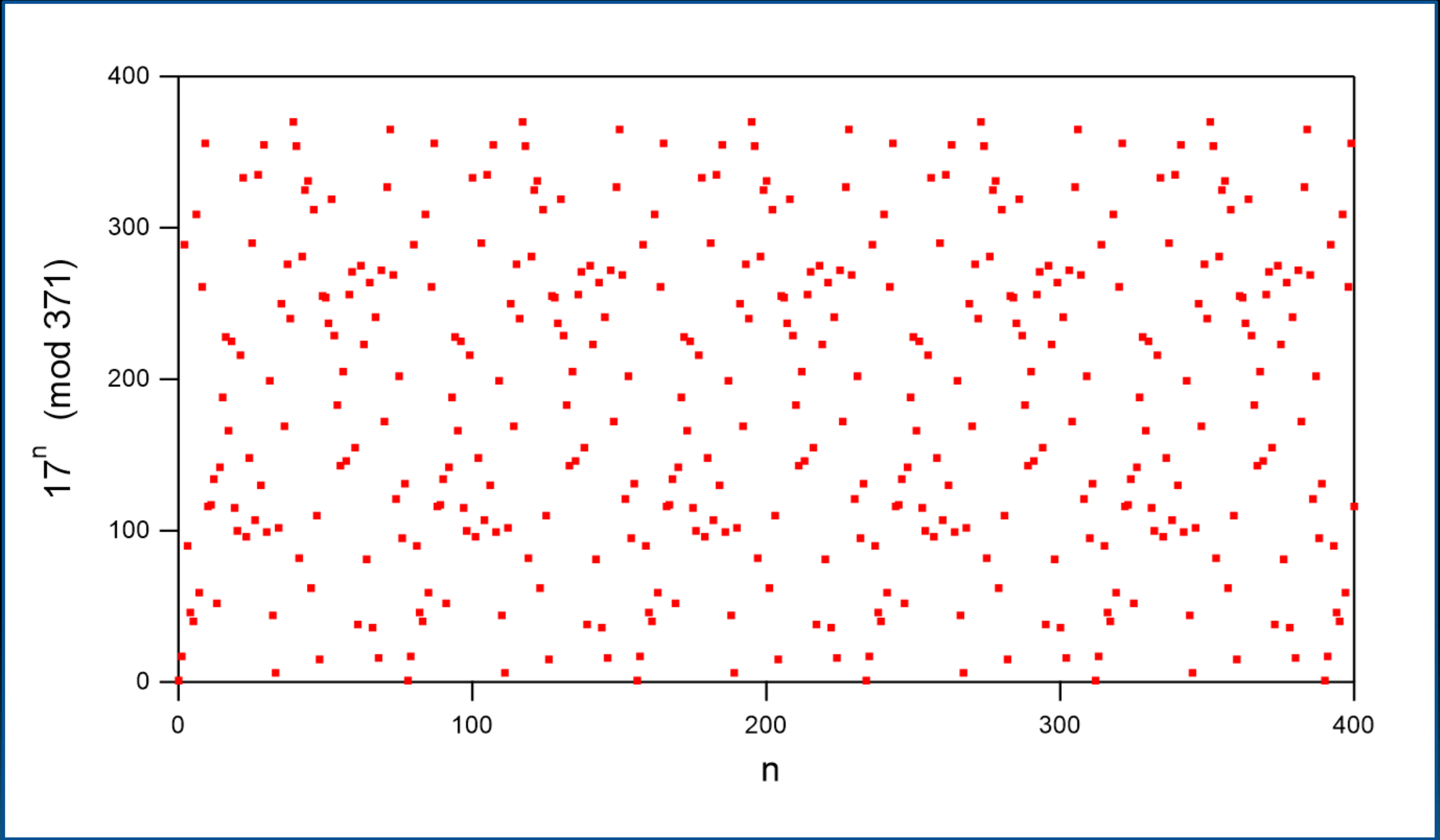
$$S_{21}^{(2)} = \{1, 2, 4, 8, 16, 11\}$$

with period/order $k = 6 = 12/2.$



Finding the period reduces to a laborious search for most N = pq

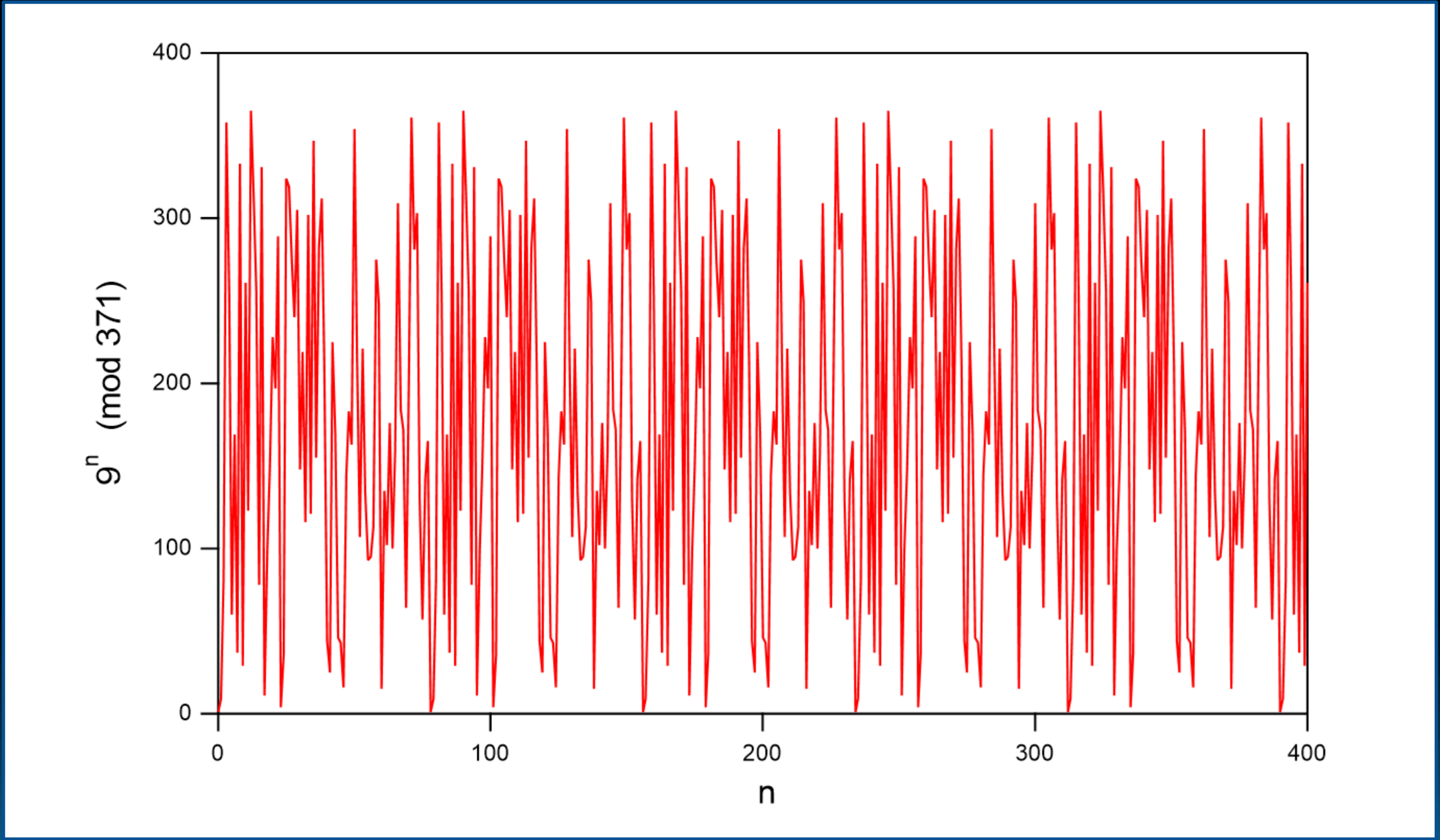
Example: $N = 371 = 7*53$; $a = 17$ $(p-1)(q-1) = 312$



The period isn't exactly obvious – but it's $k = 78 = 312/4$

Even with the dots connected the sequency sure looks random

N = 371 = 7*53; a = 9 (p-1)(q-1) = 312



You would be hard-pressed to figure out the period by calculating a few points!

An interesting observation:

$$a^k \equiv 1 \text{ (mod N)} \Rightarrow a^k - 1 \equiv 0 \text{ (mod N)}$$

If k is even (which should be roughly half the time) we can write:

$$a^k \equiv 1 \text{ (mod N)} \Rightarrow \left(a^{k/2} - 1\right)\left(a^{k/2} + 1\right) \equiv 0 \text{ (mod N)}$$

In our first example:

$$N = 21 \qquad \left(2^{6/2} - 1\right) \equiv 7 \text{ (mod N)}$$

$$a = 2 \qquad \left(2^{6/2} + 1\right) \equiv 9 \text{ (mod N)}$$

$$k_2 = 6$$

$$\left.\begin{array}{l} GCD(7, 21) = 7 \\ GCD(9, 21) = 3 \end{array}\right\}$$

These ARE the prime factors that make up N

In our second example k is again even (78):

$$a^k \equiv 1 \text{ (mod N)} \Rightarrow \left(a^{k/2} - 1\right)\left(a^{k/2} + 1\right) \equiv 0 \text{ (mod N)}$$

$N = 371$

$a = 17$

$k_2 = 78$

$$\left(17^{78/2} - 1\right) \equiv 369 \text{ (mod } 371)$$

This will never be zero

$$\left(17^{78/2} + 1\right) \equiv 0 \text{ (mod } 371)$$

Sometimes this will be!

Unlucky – we don’t learn anything from the first example.

$N = 371$

$a = 9$

$k_2 = 78$

$$\left(9^{78/2} - 1\right) \equiv 210 \text{ (mod } 371)$$

$$\left(9^{78/2} + 1\right) \equiv 212 \text{ (mod } 371)$$

$$\left.\begin{aligned} GCD(210, 371) &= 7 \\ GCD(212, 371) &= 53 \end{aligned}\right\}$$

These ARE the prime factors that make up N

f. Sub-group orders and prime factorization

Finding the order *k* of

$$a^n \text{ (mod N)} \qquad n = 0, 1, 2, 3, \ldots$$

allows you to determine the prime factors…

If…

1. *k* is even $\Longleftarrow$ 50-50!

2. $a^{k/2} \not\equiv N - 1 \text{ (mod N)} \Longrightarrow a^{k/2} + 1 \not\equiv 0 \text{ (mod N)}$ $\Longleftarrow$ >50% (See Appendix M)

$$a^k \equiv 1 \text{ (mod N)} \Rightarrow \left(a^{k/2} - 1\right)\left(a^{k/2} + 1\right) \equiv 0 \text{ (mod N)}$$

$$GCD\left(\left(a^{k/2} - 1\right), N\right) = p$$

$$GCD\left(\left(a^{k/2} + 1\right), N\right) = q$$

C. RSA Encryption

In RSA the private (decoding) key requires knowledge of two prime numbers *p* and *q* and the public (encoding) key requires knowledge of only the composite number *N* = *pq.* The fact that *finding primes* is easy, *creating a composite* is easy, and *factoring a composite* is hard is the basis for RSA encryption.

To generate the two keys Bob choose two large prime numbers *p* and *q* which he turns into a composite number *N* = pq and an encoding number (also large) *c that is relatively prime with (p-1)(q-1).*

Public Key: {N, c} Example: {91, 11} (p = 7, q = 13)

Bob creates a decoding key *d* such that

$$cd \equiv 1 \text{ (mod (p-1)(q-1)}$$

Private Key: {N, d} Example: {91, 59}

1. Encoding

Alice wants to send a private message to Bob. Bob creates a public key and sends it to Alice over a public channel.

a. Alice creates a numerical representation of the message

$\{a_1, a_2, a_3, a_4, \ldots\}$

e.g. "a" = 1, "b"=2, "c" = 3, etc.

e.g. use the ASCII codes for letters and numbers

Example: "Colby" → {3,15,12,2,25}

b. Encode the message using the pubic key to create

$\{a_1, a_2, a_3, a_4, \ldots\} \rightarrow \{e_1, e_2, e_3, e_4, \ldots\}$

$$e_i = a_i^c (\bmod N)$$

Example: Using the key: {N = 91, c = 11}

{3,15,12,2,25} → {61,85,38,46,51}

c. Send the encoded message to Bob over a public channel.

2. Decoding

Bob needs to decode the message. (In fact, anyone who can figure out the prime factors of N = 91 could do this, but factoring is hard!)

a. Decode the message using the private key to create

$\{e_1, e_2, e_3, e_4,\ldots\} \rightarrow \{m_1, m_2, m_3, m_4,\ldots\}$

$$m_i = e_i^d (\bmod \mathrm{N})$$

Example: Using the private key: {N = 91, d = 59}

$\{61,85,38,46,51\} \rightarrow \{3,15,12,2,25\}$

Which **is** the original message!

3. Decoding always works

$$\begin{aligned} m_i &\equiv e_i^d \text{ (mod N)} \\ &\equiv (a_i^c)^d \text{ (mod N)} \\ &\equiv a_i^{cd} \text{ (mod N)} \\ &\equiv a_i \text{ (mod N)} \end{aligned}$$

The proof is on page 66 of Mermin.