This will appear to behave randomly, though it can be solved for in closed form.

Ex: Find the sequence of pseudorandom numbers with $m=9$, $a=7$, $c=4$, $x_0=3$

$$x_1 = 7x_0+4 \bmod 9 = 7\cdot 3+4 \bmod 9 = 25 \bmod 9 \equiv 7$$
$$x_2 = 7\cdot 7+4 \bmod 9 = 53 \bmod 9 = 8$$
$$x_3 = 7\cdot 8+4 \bmod 9 = 60 \bmod 9 = 6$$
$$\vdots$$
$$x_8 = 7\cdot 4+4 \bmod 9 = 32 \bmod 9 = 5$$
$$x_9 = 7\cdot 9+4 \bmod 9 = 39 \bmod 9 = 3$$

Since we've reached x_0 & every term only depends on previous term $\Rightarrow$ repeats!

$3, 7, 8, 6, \ldots, 5, 3, 7, 8, 6, \ldots$

Cryptography: First classic crypto, Caesar cipher!

This method encrypts messages by shifting letters, modulo 26.

In Caesar's case he shifted by 3

$f(p) = (p+3) \bmod 26$ $\qquad$ $A \to D \quad B \to E \quad C \to F \quad \cdots \quad Z \to C$

Ex Encrypt ATTACK AT DAWN

First we can write this as numbers

0 19 19 0 2 10 $\quad$ 0 19 $\quad$ 3 0 22 13

Shift by 3:

3 22 22 3 5 13 $\quad$ 3 22 $\quad$ 6 3 25 16

Convert back:

DWWDFN $\quad$ DW $\quad$ GDZQ

If we are given a message to decrypt it we must subtract the shift ~~[illegible]~~ modulo 26.

Ex: Decrypt DEZA RWZMLW HLCXTYR if it was encrypted with shift = 11.

First write in numbers:

3 4 25 0 17 22 25 12 11 22 7 11 2 23 19 24 17

Now subtract 11: (or add 26 - 11 = 15)

18 19 14 15 6 11 14 1 0 11 22 0 17 12 8 13 6

S T O P G L O B A L WA R M I N G

This method generalizes: we can define $f(p) = (a \cdot p + k) \mod 26$

This method of decrypting works as long as a^{-1} exists mod 26

which means $\gcd(a, 26) = 1$.

This is called an affine cipher. If $\gcd(a, 26) \neq 1$ decryption is very different.

Ex: Decrypt NJLNRBN DBJNTDNPJJJ

where $a = 2$, $k = 1$.

First in numbers: 13 9 11 13 17 1 13 3 1 9 13 19 3 13 15 9 9 9

Subtract 1: 12 8 10 12 16 0 12 2 0 8 12 18 2 12 14 8 8 8

But now we have a problem can't compute $2^{-1} \mod 26$

Dividing by 2 is difficult! $12 \to 6$ or $12 \to 19$ $(19 \cdot 2 = 38 \equiv 12 \mod 26)$

So we have to look at both options:

Divide each number by 2 (assume no wrap)

and add 26 & divide by 2 (assume wrap)

Then we get:

no wrap: G E F G I A G B A E G J B G H E E E

wrap: T R S T V N T O N R T W O T U R R R

neither of these make a message! what to do now?

The Examples we've ~~we~~ seen so far are block ciphers, meaning exactly as many characters they take in (1 in our case) they spit out.

Block ciphers get much more complicated: can work with pairs of letters such as with Playfair or with a fixed number of bits as with AES.

All of the ciphers we have talked about so far are easily broken, via cryptanalysis. we've essentially just swapped some letters for other letters: that is a letter always encrypts to the same letter.

This allows <u>frequency analysis</u>. Attackers can exploit the fact that in (unencrypted) ~~En~~ English E shows up 13% of the time T 9% A 8% O 8% etc

we can do the same analysis on encrypted messages! If Z shows up 12% of the time Z = E probably, etc.

Cryptography in the modern world is more complicated.

We use a number of tools: In particular public-key cryptography.

The previous ciphers have been private key (or Symmetric) encryption

Meaning the communicating parties must agree on a key ahead of time.

I the real world this is difficult, how do you ensure no-one else gets it?

or if you can safely get the key between you, why not use that method to

Share messages?

To address this the idea of public key crypto was developed. Instead of

every pair of ppl needing a private key we will allow every single person to

have a public & private key.

Enter RSA (Rivest, Shamir, Adleman):

Here every user has two keys: a public (encryption key) $(n, e) \in \mathbb{Z} \times \mathbb{Z}_n$

where $n = p \cdot q$ $p, q \in \mathbb{Z}$ prime. e relatively prime to $(p-1)(q-1)$.

Given two (very large) primes ($p, q \sim 2^{2048}$) it is (relatively) simple

to compute n. But just given $n \sim 2^{4096}$ it is very difficult

to compute p, q.

To encrypt a message M, we must first turn it into an integer (mod n)

This can be done many ways simplest is as before $A \to 00$ $B \to 01$ ... $Z = 25$

Then concatenate So HELLO would become

0704 11 11 14

If Necessary we split a message into blocks ~~that~~ are not bigger than n.
To avoid small numbers we may pad our message with dummy X's (23's)

Thus $M \to (m_1, m_2, \ldots, m_k) \in \overbrace{\mathbb{Z} \times \mathbb{Z} \cdots \times \mathbb{Z}}^{k \text{ times}}$

Then our cipher text is computed as $c_i = m_i^e \bmod n$. (remember we know how to do this quickly!)

Ex: Encrypt STOP with key $(2537, 13)$

STOP = 18 19 14 15 note $18191415 > 2537$

So split into blocks of 2: 1819, 1415 no room for padding

$1819^{13} \bmod 2537 = 2081$ $1415^{13} \bmod 2537 = 2182$

$\to$ 2081 2182 is our cipher text.

RSA Decryption To decrypt messages we need a decryption key.

This is $d \equiv e^{-1} \bmod (p-1)(q-1)$ note if I choose p, q, e then I can compute d since $\gcd(e, (p-1)(q-1)) = 1$.

Note: $de = 1 + k(p-1)(q-1)$ for some $k \in \mathbb{Z}$.

$$\Rightarrow c_i^d = (m_i^e)^d = m_i^{de} \equiv m_i^{1 + k(p-1)(q-1)} \bmod n$$
$$= m_i \cdot \left(m^{(p-1)}\right)^{k(q-1)} \bmod n \quad \text{(Fermats little theorem)}$$
$$\equiv m_i \cdot 1^{k(q-1)} \bmod n$$
$$\equiv m_i$$

Ex: Decrypt 0981 0461 if our public key is the same as above.

$n = 43 \cdot 59 = 2537 \quad e = 13$

need d s.t. $d \equiv e^{-1} \bmod 2436 = (p-1)(q-1)$

$2436 = q \cdot e + r \qquad 1 = \cdots$

$\vdots$

$+1 \qquad \therefore d = 937.$

$M = C^{937} \bmod 2537.$

$0981^{937} \bmod 2537 = 0704$

$0461^{937} \bmod 2537 = 1115$

$07041115 = \text{HELP}.$

why does this work? multiply integers is fast: two k-digit numbers can be multiplied in k^2 steps. Find d from e, n is Extended Euclidean Alg (also fast).

However there is no fast method for factoring integers it is an NP problem

remember what happened last time I talked about an NP problem?

while there is no polynomial time algorithm there are algorithms that aren't horribly long, for small integers.

This was an example of a public key system, allowing ppl who can't share a key to communicate. To send a message to Bob Alice will encrypt her message under his key, so he can decrypt it.

Diffie Hellman Key Exchange ; Very popular Algorithm to share keys

(1) Allice & Bob agree to use $p \in \mathbb{Z}$ as their public prime and chose particular $a \in \mathbb{Z}_p$.

(2) Alice chooses $\overset{\text{secret}}{\vee}$ $k_1 \in \mathbb{Z}_p$ Sends $a^{k_1} \bmod p \rightarrow$ Bob

(3) Bob chooses secret $k_2 \in \mathbb{Z}_p$ Sends $a^{k_2} \bmod p \rightarrow$ Alice

(4) Alice Computes $(a^{k_2})^{k_1} \bmod p$

(5) Bob Computes $(a^{k_1})^{k_2} \bmod p$.

Now Alice & Bob share $a^{k_1 k_2} \bmod p$!

The security of this is on the difficulty of the Discrete Log Problem (also NP) given $a^k \bmod p$, a, p finding k is very difficult.

Public key crypto is fun & does awesome math stuff, but it's very slow. Thus realistically it can't be used much. The way most crypto (internet) works is

ECDHE_RSA - to share keys & verify server is who it says it is.

AES_128_GCM - to encrypt messages (private key = fast) & GCM for verification.