1. Traditional Encryption

- Traditional cryptography involves a function T (for example, substitute each letter for the kth letter beyond that one)
- Deciphering the message is very easy, using the function T^{-1} (the inverse of T, which will be substituting each letter for the kth letter after that one)
- If the algorithm becomes public knowledge (or even if it does not), it would not take a cryptanalyst long to decrypt a message by testing every possible value of k

2. Public Key Encryption

- In public key cryptography, the public key, k need never be changed
- The receiver of messages has a private key, k'
- There is a relationship between k and k' such that k' is very hard to compute given k
- But k is very easy to compute given k' (so new public keys can be created easily)

The Subset Sum Problem

- This problem is **NP-Complete**
- I.e. there is no algorithm that is able to solve the subset problem in polynomial, $O(n^k)$ time
- So a solution can only be produced by sometimes waiting an extraordinarily long time – most likely a significant proportion of the 2^n possible solutions will have to be tested
- This forms the basis of the Diffiie-Hellman Key Exchange

Encrypting a Message

- Each user is given a public key made up of n positive integers, a_1, a_2,..., a_n
- The message x is transformed into a string of binary digits, which are partitioned into blocks of length n, $(x_1, x_2, \dots, x_n)$

The final output is $B_x = \sum_{i=1}^{n} x_i a_i$

An Example of Encryption

- Take the word, SECRET, encoded in 7-bit ASCII:
- Taking $n = 7$ (although much higher values are usually used), and using $(901,568,803,39,450,645,1173)$ as the key, encoding the S will give:

$$1 \times 901 + 0 \times 568 + 1 \times 803 + 0 \times 39 + 0 \times 450 + 1 \times 645 + 1 \times 1173 \Rightarrow B_x = 3522$$

- Anybody attempting to decipher the message, even with the knowledge of what public key integers were used, will likely have to attempt a significant proportion of the $2^7 = 128$ possible combinations (for each letter!) – and n will normally be very much larger than 7

Decrypting the Message

- The receiver of the message uses a private key, $(a'_1, a'_2, \dots, a'_n)$ and two integers, w and m to decrypt the message
- The public key is related to the private key such that:

$$a_i = (w \times a'_i) \bmod m$$

- Hence to calculate the message bits, x_i, we use a special version of the subset sum problem, such that we have a subset of $(a'_1, a'_2, \dots, a'_n)$ that gives B'_x, where $B'_x = (B_x \times w^{-1}) \bmod m$

(Here we take w^{-1} to be the inverse of w in the field of integers modulo m – i.e. $w \times w^{-1} = 1 \bmod m$)

- This is really just the inverse of the encryption process, T^{-1}, and happens to be very easy to solve quickly (in linear $O(n)$) (given that the private key is superincreasing – each integer is larger than the sum of integers preceding it)
  - Take the largest integer, a'_n: if $B'_x > a'_n$, then include a'_n, else discard it
  - Take the next largest integer, a'_i, if $B'_x > \sum a'_{included} + a'_i$, then include a'_i, else discard it
  - Repeat the second step until $\sum a'_{included} = B'_x$

An Example of Decryption

- Taking the example from earlier, we already have:

  $B_x = 3522$ and $k = (901,568,803,39,450,645,1173)$
- Now we let, for example:

  $w = 901$, $m = 1234$ and hence $k' = (1,2,5,11,32,87,141)$ (since $a_i = (w \times a'_i) \bmod m$)
- This means we have to find the set of a'_i such that $B'_x = (3522 \times (901)^{-1}) \bmod 1234 = (3522 \times 1171) \bmod 1234 = 234$
- Applying the algorithm from the previous slide to $B'_x = 234$ and $k' = (1,2,5,11,32,87,141)$, we get:

  $141 + 87 + 5 + 1 = 234$, giving (1,0,1,0,0,1,1), which is the 7-bit ASCII for 'S'

An Overview of How it Works

- We can prove that this method always works nicely by analysing the algebra of T and T^{-1}:

T is obviously given by $B_x = \sum_{i=1}^{n} x_i a_i$ (which is equivalent to $\sum_{i=1}^{n} x_i \times w a'_i \bmod m$)

Meanwhile, T^{-1} is given by $B'_x = (B_x \times w^{-1}) \bmod m$,

which gives $B'_x = \sum_{i=1}^{n} x_i \left(w a'_i \bmod m\right) w^{-1} \bmod m$ and hence $B'_x = \sum_{i=1}^{n} x_i a'_i$

I.e. the message x encoded in B_x as $\sum_{i=1}^{n} x_i a_i$ is encoded in B'_x as $\sum_{i=1}^{n} x_i a'_i$

How to Break it?

- There are only two points of vulnerability (short of the private key being discovered), which are:

  (a) An algorithm is discovered that can solve NP-Complete problems quickly (most likely impossible), or

  (b) The problem used isn't actually an NP-Complete problem

- This is precisely what happened to the Diffie-Hellman-Merkle system in the 1980s – since the "public" subset-sum problem is very much a special case (rather than a general version of the subset-sum problem (which is an NP-Complete problem)), it is solvable in polynomial time

- There are, however, other forms of public-key encryption, which instead rely upon the general case of an NP-complete problem (such as the RSA cryptosystem (which instead uses the prime factorization problem) which is currently solvable in $O(e^{(\log n \log\log n)^{\frac{1}{2}}})$ (sub-exponential time) steps)

- However, even the RSA cryptosystem is solvable in polynomial time given a quantum computer with enough qubits to perform Shor's Algorithm (which simply reduces the prime-factorization problem to an order-solving one) – 21 in 2012 (and 56153 in 2014 but with a different algorithm)