Integrating Quantum Concepts into Cyber Security

Session 3: Algorithms

Dr William Joseph Spring

ACSAC 35, Condado Plaza Hilton, San Juan, Puerto Rico, USA

9th – 13th December 2019

Key Agreement Protocols

- Diffie Hellman Key Agreement Protocol
- BB84
- B92

Quantum Cryptography

Key Agreement Protocols

- The Diffie Hellman Key Agreement Protocol is a classical based protocol
  - Uses a multiplicative cyclic group, a primitive and the DLP to agree a symmetric key, the same key for sender and receiver
- BB84, B92 and E91 are quantum based key agreement protocols
- They employ:
  - No cloning Theorem (Quantum Property)
  - Information gain implies disturbance (Quantum Property)
  - Information Reconciliation (Classical Technique)
  - Privacy amplification (Classical Technique)

Shors Algorithm

In the mid 1990's Peter Shor published a paper in which he established that given a quantum computer of sufficient processing power his algorithm would break any scheme based on either the IFP or the DLP

It broke the following algorithms in a very efficient manner

- Diffie Hellman
- RSA
- El Gamal – all three forms

And led to the problem of what to replace these algorithms with. These were/are very efficient in comparison to current alternatives.

The Diffie Hellman Key Agreement Protocol

- This is a classical key agreement protocol, in contrast to a key exchange protocol, since neither sender nor receiver know the key until the end of the protocol.
- The protocol is based upon the discrete logarithm problem and hence is based upon a finite multiplicative cyclic group, generated by a primitive element . The primitive generates the integers 1, 2, 3, ... , p-1, where p is an agreed prime number. is often referred to as a primitive root of p, by which we mean that generates each of the integers from 1 up to p-1.

The Diffie Hellman Key Agreement Protocol

- This is a classical key agreement protocol, in contrast to a key exchange protocol, since neither sender nor receiver know the key until the end of the protocol.
- The protocol is based upon the discrete logarithm problem and hence is based upon a finite multiplicative cyclic group, generated by a primitive element . The primitive generates the integers 1, 2, 3, ... , p-1, where p is an agreed prime number. is often referred to as a primitive root of p, by which we mean that generates each of the integers from 1 up to p-1.

The Diffie Hellman Key Agreement Protocol

We recall the discrete logarithm problem:

- given a prime number p,
- a primitive α,
- and $\beta \in < \alpha > = \{1,\ 2,\ 3,\ \ldots\ ,\ p-1\}$ (the cyclic group generated by α)
- find r such that $\beta = \alpha^r \bmod p$

Note r is integer valued, $1 \leq r \leq p-1$

Example

$\alpha = 2, p = 13$

The Diffie Hellman Key Agreement Protocol

The protocol runs as follows:

Alice	Bob
1. Select the primitive α . Bob and Alice agree the primitive before starting the protocol	1. Select the primitive α . Bob and Alice agree the primitive before starting the protocol
2. Selects a secret random number *a*	2. Selects a secret random number *b*
3. Sends α^a to Bob	3. Sends α^b to Alice
4. Alice receives α^b from Bob and raises this to the power a to obtain the agreed key $\alpha^{ba} \bmod p$	4. Bob receives α^a from Bob and raises this to the power b to obtain the agreed key $\alpha^{ab} \bmod p$

Alice and Bob now posses the same key which they can now use in a symmetric key cryptography scheme such as (for example) DES (the Data Encryption Scheme) or AES (the Advanced Encryption Scheme).

Quantum Key Agreement

- BB84
- B92
- E91

BB84

1. Alice chooses (4 +δ)*n* random data bits *a* where *n* >> m the length of the sought key

2. Alice chooses a (4 +δ)*n* random - bit string *b*.

She encodes each data bit as {|0>, |1>} if the corresponding bit of *b* is 0 or {|+>, |->} *b* is 1

3. Alice sends the resulting state to Bob

4. Bob receives the (4 +δ)*n* qubits, announces this fact, and measures each qubit in the X or Z basis at random

5. Alice announces b

6. Alice and Bob discard any bits where Bob measured a different basis than Alice prepared. With high probability there are at least 2n bits left (if not abort the protocol). They keep 2n bits.

BB84

7. Alice selects a subset of n bits that serve as the ‘check bits’, a check on Eves interference, and tells Bob which bits were selected

8. Alice and Bob announce and compare the values of the n ‘check bits’. If more than an acceptable number disagree then they abort the protocol

9. Alice and Bob perform information reconciliation and privacy amplification on the remaining *n* bits to obtain *m* shared key bits

B92

We consider what happens to one bit at a time. Generalisation to a block follow naturally as with BB84

1. Alice prepares one random classical bit *a,* and depending upon the result sends Bob

$$|\psi\rangle = \begin{cases} |0\rangle & \text{if a} = 0 \\ |+\rangle & \text{if a} = 1 \end{cases}$$

2. Depending upon the random classical bit a' that Bob generated, Bob uses either the Z or X basis and obtains his result b - which is either a 0 or a 1

B92

3. Bob announces *b* but keeps *a'* secret

4. Alice and Bob conduct a public discussion keeping only those pairs {*a, a'*} for which *b = 1*. Note that when *a = a',* then *b = 0.* Only when *a' = 1 – a* does *b = 1* and this occurs with probability ½

5. The final key is *a for Alice and 1 – a' for Bob*

E91

See The Ekert Protocol by Nina Nikolina Ilic:

http://www.ux1.eiu.edu/~nilic/Nina's-article.pdf

and

Ekert's original paper: 'Quantum cryptography based on Bell's theorem', Physical Review Letters, vol. 67, no. 6, 5 August 1991, pp. 661 - 663.

http://prola.aps.org/pdf/PRL/v67/i6/p661_1

Activity 3a – Key Agreement

- Diffie Hellman Key Agreement Protocol
- BB84
- B92
- E91

Shors Algorithm

Areas for Discussion

Shors Algorithm

- Hidden Subgroup Problem
- A Cyclic Relationship for RSA
- The Quantum Fourier Transform
- Shors Algorithm
- Shors Algorithm and the DLP

Introduction

Shor's Algorithm

This has the potential to make any public key cryptography algorithm based on either the DLP or the IFP redundant.

- No more RSA, DSA, El-Gamal, ...
- Related to the Hidden subgroup problem for Abelian groups

Grovers Algorithm

A search algorithm offering potential benefits over classical alternatives.

Is this the end of symmetric key cryptography?

The Hidden Subgroup Problem

The Hidden Subgroup Problem may be stated as follows:

Input: A finite abelian group G and a function

$$\rho : G \to R$$

where R is a finite set

Promise: There exists a nontrivial subgroup H of G such that $\rho : G \to R$ is constant and distinct on each coset of H

Output: A generating set for H

A Cyclic Relationship for RSA

Plaintext and Ciphertext

- Let $n = pq$ with $p, q \in \mathbb{N}^+$, distinct odd prime numbers so $p \neq q$.
- Let M denote the set of all possible plaintext messages m and C denote the set of all corresponding ciphertext messages c.
- From a cryptanalysts point of view the problem of finding p and q can be re-expressed in terms of finding the order r of an element in C (and correspondingly M).
- If we can find the order then we can find (n, p, q, a, b) and hence break the RSA algorithm

A Cyclic Relationship for RSA

Lemma 1

The order of an element $c \in C$ and its corresponding $m \in M$ are equivalent in $\mathbb{Z}_{\phi(n)}$, with $\phi(n) = (p-1)(q-1)$ Eulers Totient function. We note that these numbers are relatively prime to p and q and hence they are relatively prime to n.

Proof

$c = m^b \bmod n \quad \Rightarrow \quad \mathrm{c} \in < m > \quad$ the cyclic group generated by m

$\Rightarrow \quad \forall i \in \mathbb{N}^+, \quad \mathrm{c}^i \in < m > \quad \Rightarrow \quad < c > \quad \leq \quad < m >$

by which we mean $< c >$ is a subgroup of $< m >$. Likewise

$m = c^a \bmod n \quad \Rightarrow \quad \mathrm{m} \in < c > \quad \Rightarrow \quad < m > \quad \leq \quad < c >$

$\Rightarrow$ m and c have the same order

A Cyclic Relationship for RSA

Lemma 2

Let (n, b) denote Bobs public RSA key.

Let r denote the order of $c \in C$.

Then $i)$ $\exists\, d \in \mathbb{Z}_r$ such that $bd = 1 \bmod r$

and ii) $m = c^d$ in $\mathbb{Z}_n$

So given the Bob's public key (n, b), if we can find r, then we can find d and hence obtain the plaintext value m.

Hence the problem of breaking RSA may be viewed as the problem of finding the order of the ciphertext vaue c.

In other words, solving the problem of finding the order of $f(x) = c^x \bmod n$

Euclids Algorithm

To find the gratest common divisor of a and b,

gcd(a,b) with a > b > 0, we may use Euclids algorithm

Euclid(a,b)

1. $A \leftarrow a;\ B \leftarrow b$
2. if $B = 0$ return $A = \gcd(a, b)$
3. $R = A \bmod B$
4. $A \leftarrow B$
5. $B \leftarrow R$
6. go to 2

Euclids Algorithm

Example

gcd(63, 35)

63 = 35x1 + 28

35 = 28x1 + 7

28 = 7x4 + 0

gcd(63, 35) = 7

Example

We can go further and obtain p and q, where n =pq.

The following simple example illustrates the concept for factorising n = 21.

Let $f(x) = c^x \bmod n$ and $c \in \mathbb{Z}_n$. The procedure is successful if r is even and $a^{r/2} \neq -1 \bmod n$.

Method:

1. Select c coprime to n.
2. Let $f(x) = c^x \bmod n$. Compute $f(1)$, $f(2)$, $f(3)$,...
3. Establish the period.
4. Calculate $\gcd(a^{r/2} \pm 1, n)$ using Euclids algorithm.

A Simple Example

1. For c coprime to $n = 21$, $c \in \{2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$

no common factors for c and n except 1. Take $c = 2$ for example.

2. Then $f(1) = 2^1 \bmod 21 = 2$

$$f(2) = 2^2 \bmod 21 = 4$$

$$f(3) = 2^3 \bmod 21 = 8$$

$$f(4) = 2^4 \bmod 21 = 16$$

$$f(5) = 2^5 \bmod 21 = 32 \bmod 21 = 11$$

$$f(6) = 2^6 \bmod 21 = 2 \times 11 \bmod 21 = 1$$

$$f(7) = 2^7 \bmod 21 = 2 \times 1 \bmod 21 = 2 = f(1)$$

3. Hence the period $r = 6$ and we have $f(x) = f(x + r)$ for $r = 6$.

A Simple Example

4. $\gcd(a^{r/2} \pm 1, n) = \gcd(2^{6/2} \pm 1, 21)$

$= \gcd(8 \pm 1, 21)$

$= \gcd(9, 21)$ and $\gcd(7, 21)$

$= 3$ and 7

Note r is even and $a^{r/2} = 2^{6/2} = 8 \neq -1 \bmod 21$ and the method works.

In the following example the method doesn't work:

A Simple Example

1. For c coprime to $n = 21$, $c \in \{2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$

no common factors for c and n except 1. Take $c = 5$ for example.

2. Then $f(1) = 5^1 \bmod 21 = 5$

$$f(2) = 5^2 \bmod 21 = 25 \bmod 21 = 4$$

$$f(3) = 5^3 \bmod 21 = 125 \bmod 21 = 20$$

$$f(4) = 5^4 \bmod 21 = 5 \times 20 \bmod 21 = 16$$

$$f(5) = 5^5 \bmod 21 = 5 \times 16 \bmod 21 = 17$$

$$f(6) = 5^6 \bmod 21 = 5 \times 17 \bmod 21 = 1$$

$$f(7) = 5^7 \bmod 21 = 5 \times 1 \bmod 21 = 5 = f(1)$$

3. Hence the period is 6

A Simple Example

4. $\gcd(a^{r/2} \pm 1, n) = \gcd(5^{6/2} \pm 1, 21)$

$= \gcd(125 \pm 1, 21)$

$= \gcd(124, 21)$ and $\gcd(126, 21)$

$= ???$

Here r is even but $a^{r/2} = 125 == 6 \times 21 - 1 = -1 \bmod 21$

and the method doesn't work.

The Quantum Fourier Transform

A discussion on the QFT and its relationship to the nth roots of unity

The Quantum Fourier Transform for n qubits is given by the mapping:

$$\left|x\right\rangle_n \mapsto \frac{1}{2^{n/2}} \sum_{y=0}^{n-1} e^{2\pi i x y / 2^n} \left|y\right\rangle_n$$

The transformation is a unitary transformation

We consider the case of 1 qubit and 2 qubits to illustrate the relationship.

The Quantum Fourier Matrix

Let $\omega^k = e^{i2\pi k/n}$ denote the nth root of unity for $1 \le k \le n$

We define the Quantum Fourier Matrix of order n to be:

$$\frac{1}{\sqrt{n}}\begin{bmatrix} 1 & 1 & 1 & 1 & . & . & . & 1 \\ 1 & \omega & \omega^2 & \omega^3 & . & . & . & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & . & . & . & \omega^{2}_{n-2} \\ 1 & \omega^3 & \omega^6 & \omega^9 & . & . & . & \omega^{n-3} \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ 1 & \omega^{n-1} & \omega^{n-2} & \omega^{n-3} & . & . & . & \omega \end{bmatrix}$$

R. J. Lipton, K. W. Regan, *Quantum Algorithms via Linear Algebra,* MIT Press, 2014

M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information,* CUP, 2010

Shors Algorithm

Discussion

Shors Algorithm consists of two parts:

1. A classical part
   - We find gcd(a, N) for a in $\mathbb{Z}_N$.
   - If the gcd is 1 we use the quantum part. If not then we have a factor
2. A quantum part
   - We start with n input values and n_0 output values each set to $|0\rangle$
   - Apply Hadamard's to each of the n input values. This will give us n $|+\rangle$ values resulting in

$$\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle_n |0\rangle_{n_0}$$

This is further initialised to

$$\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle_n | f(x)\rangle_{n_0}$$

Shors Algorithm

- The next step is to measure the output states
  - If we obtain $|f(x_0\rangle$ then from Borns Rule we obtain

$$|\psi\rangle_n = \frac{1}{\sqrt{m}} \sum_{k=0}^{m-1} |x_0 + kr\rangle_n$$

with $0 < x_0 < r$ and m-1 is the maximum value such that $x_0 + (m-1)r < 2^n$. m will be the integer value associated with $2^n/r$ or $1+ 2^n/r$

  - Apply the QFT to the input value $|x_0 + kr\rangle_n$ to obtain

$$|x_0 + kr\rangle_n \mapsto \frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} \frac{1}{\sqrt{m}} \sum_{k=0}^{m-1} e^{2\pi i(x_0+kr)y/2^n} |y\rangle_n = \frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} \frac{1}{\sqrt{m}} \left(\sum_{k=0}^{m-1} e^{2\pi ikry/2^n}\right) |y\rangle_n$$

Shors Algorithm

- The next step is to measure the output from the QFT to obtain

$$p(y) = \left| \frac{1}{2^n m} \sum_{k=0}^{m-1} e^{2\pi i k r y / 2^n} \right|^2$$

  - Finding the maxima of p(y) leads us to an estimate for r which we can check to see if it satisfies f(x) = f(x + r')

  - If it works then we have the period.
  - If not test for multiples of r' or values near y

  - If unsuccessful then retry from the beginning of the quantum part

Significance of Shors Algorithm

The **significance** of Shors Algorithm lies in the possibility for a cryptanalytic attack that if implemented would lead to the end of asymmetric cipher schemes such as RSA and El Gamal, based on either the integer factorisation problem or the discrete logarithm problem.

It should be **noted** that although the potential threat is perceived as real and significant there are counterclaims suggesting that it will not be possible to implement Shors algorithm beyond four qubits in 3 dimensions.

See for example: http://arxiv.org/abs/1502.05926 and https://www.lightbluetouchpaper.org/2015/02/23/maxwell/

PQC/Quantum Free Algorithms

Post Quantum Cryptography/Quantum Free Algorithms

- The potential threat from Shors algorithm has led to a concerted effort to find classical algorithms thought to be safe from quantum based attacks. Potential Candidates for this include:
  - Hash-based cryptography
    - The classic example is Merkle's hash-tree public-key signature system (1979), building upon a one-message-signature idea of Lamport and Diffie.
  - Code-based cryptography
    - The classic example is McEliece's hidden Goppa-code public-key encryption system (1978).
  - Lattice-based cryptography
    - The example that has perhaps attracted the most interest, not the first example historically, is the Hoffstein– Pipher–Silverman "NTRU" public-key-encryption system (1998).
  - Multivariate-quadratic-equations cryptography
    - One of many interesting examples is Patarin's "HFEv–" public-key-signature system (1996), generalizing a proposal by Matsumoto and Imai.
  - Secret-key cryptography
    - The leading example is the Daemen–Rijmen "Rijndael" cipher (1998), subsequently renamed "AES," the Advanced Encryption Standard

Daniel J Bernstein, Post-Quantum Cryptography, Springer, 2008
https://pqcrypto2020.inria.fr/

Activity 3b – QFT, Shors & PQC/Quantum Free Algorithms

Tutorial 3b

1. **Using the approach taken in the ‘Simple Example’ find the factors of n = 15.**
2. **Establish the eighth roots of unity.**
3. **What would be the quantum Fourier transform matrix for three qubits?**

Appendix – IFP, RSA; DLP, El Gamal

The Integer Factorisation Problem

Given n a positive integer, there exists primes p_i such that $n = a p_1^{e1} p_2^{e2} p_3^{e3} \ldots P_r^{er}$ in which a is a unit.

Examples

$72 = 2 \times 36 = 2 \times 2 \times 18 = 2 \times 2 \times 2 \times 3 \times 3 = 2^3 \times 3^2$

One application of the integer factorisation problem is the Rivest Shamir Adleman algorithm using n = pq with p and q primes chosen such that factorisation for n is unlikely given current available techniques and technology.

RSA and El Gamal

RSA: Here ‘Bobs key for the entire system’ is given by a 5-tuple (n, p, q, a, b) with n = pq, and ab = 1 mod(p-1)(q-1).

Each participant (Alice and Bob) have a public and a private pair of keys, each being the inverse of the other.

Bob’s public key is given by the pair (n, a), this is made available to all and Alice sends an encrypted message to Bob via $e(m) = m^a \bmod n = c$.

Bobs private key is (p, q, b), with decryption of c given by $d(c) = c^b \bmod n = m$

RSA and El Gamal

For the **DLP** we have the **El Gamal algorithm**.

The '**key**' for this cryptographic scheme is given by

$(\alpha, \beta, \mathrm{r}, p, k)$ in which $(\alpha, \beta, \mathrm{r}, p)$ are obtained from the DLP and k is a **mask** generated by the sender, Alice.

In order for Alice to send an encrypted message to Bob, Alice uses Bobs **public key** (α, β, p) together with the mask k, generated by Alice to obtain the ciphertext pair

$$e(m,k) = (\alpha^k \bmod p, m\beta^k \bmod p) = (\mathrm{c}_1, \mathrm{c}_2)$$

For decryption Bob uses the private key r

$$d(c_1, \mathrm{c}_2) = \mathrm{c}_2\, c_1^{-r} \bmod p = m$$

Rivest, Shamir and Adleman

Different to DES

- DES
  - Symmetric Key
  - Product Cipher
- RSA
  - Asymmetric (Public) Key
  - One way function – large computational power
  - 512 bit key – DES less expensive to compute
  - 512 soon factorable, 768 and 1024 bit keys used

Public / Private Key Algorithm

1. Choose p and q, two prime numbers such that each is approximately 256 bits long
2. Let n = pq
3. Choose encryption key e such that e is relatively prime to $(p-1)(q-1)$
4. Choose decryption key d such that

$$d = e^{-1} \bmod((p-1)(q-1))$$

5. Public key is (e,n), Private Key is (d, n)
6. p and q are not disclosed

Encryption / Decryption

Public Key = (e, n) and Private Key = (d, n)

- Let m denote the plaintext message
- Let c denote the resulting ciphertext

For encryption we use the formula:

$$c = m^e \bmod n$$

For decryption we use the formula:

$$m = c^d \bmod n$$

Example

We consider the following example:

Let p = 7 and q = 11

Then $n = pq = 7 \times 11 = 77$

and

$(p-1)(q-1) = (7-1)(11-1) = 6 \times 10 = 60$

So we need to pick a value of e that is relatively prime to 60

Mod 60

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
2	0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40	42	44	46	48	50	52	54	56	58	0	2	4	6	8	10
3	0	3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51	54	57	0	3	6	9	12	15	18	21	24	27	30	33	36	39	42	45
4	0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	0	4	8	12	16	20
5	0	5	10	15	20	25	30	35	40	45	50	55	0	5	10	15	20	25	30	35	40	45	50	55	0	5	10	15	20	25	30	35	40	45	50	55
6	0	6	12	18	24	30	36	42	48	54	0	6	12	18	24	30	36	42	48	54	0	6	12	18	24	30	36	42	48	54	0	6	12	18	24	30
7	0	7	14	21	28	35	42	49	56	3	10	17	24	31	38	45	52	59	6	13	20	27	34	41	48	55	2	9	16	23	30	37	44	51	58	5
8	0	8	16	24	32	40	48	56	4	12	20	28	36	44	52	0	8	16	24	32	40	48	56	4	12	20	28	36	44	52	0	8	16	24	32	40
9	0	9	18	27	36	45	54	3	12	21	30	39	48	57	6	15	24	33	42	51	0	9	18	27	36	45	54	3	12	21	30	39	48	57	6	15
10	0	10	20	30	40	50	0	10	20	30	40	50	0	10	20	30	40	50	0	10	20	30	40	50	0	10	20	30	40	50	0	10	20	30	40	50
11	0	11	22	33	44	55	6	17	28	39	50	1	12	23	34	45	56	7	18	29	40	51	2	13	24	35	46	57	8	19	30	41	52	3	14	25
12	0	12	24	36	48	0	12	24	36	48	0	12	24	36	48	0	12	24	36	48	0	12	24	36	48	0	12	24	36	48	0	12	24	36	48	0
13	0	13	26	39	52	5	18	31	44	57	10	23	36	49	2	15	28	41	54	7	20	33	46	59	12	25	38	51	4	17	30	43	56	9	22	35
14	0	14	28	42	56	10	24	38	52	6	20	34	48	2	16	30	44	58	12	26	40	54	8	22	36	50	4	18	32	46	0	14	28	42	56	10
15	0	15	30	45	0	15	30	45	0	15	30	45	0	15	30	45	0	15	30	45	0	15	30	45	0	15	30	45	0	15	30	45	0	15	30	45
16	0	16	32	48	4	20	36	52	8	24	40	56	12	28	44	0	16	32	48	4	20	36	52	8	24	40	56	12	28	44	0	16	32	48	4	20
17	0	17	34	51	8	25	42	59	16	33	50	7	24	41	58	15	32	49	6	23	40	57	14	31	48	5	22	39	56	13	30	47	4	21	38	55
18	0	18	36	54	12	30	48	6	24	42	0	18	36	54	12	30	48	6	24	42	0	18	36	54	12	30	48	6	24	42	0	18	36	54	12	30
19	0	19	38	57	16	35	54	13	32	51	10	29	48	7	26	45	4	23	42	1	20	39	58	17	36	55	14	33	52	11	30	49	8	27	46	5
20	0	20	40	0	20	40	0	20	40	0	20	40	0	20	40	0	20	40	0	20	40	0	20	40	0	20	40	0	20	40	0	20	40	0	20	40
21	0	21	42	3	24	45	6	27	48	9	30	51	12	33	54	15	36	57	18	39	0	21	42	3	24	45	6	27	48	9	30	51	12	33	54	15
22	0	22	44	6	28	50	12	34	56	18	40	2	24	46	8	30	52	14	36	58	20	42	4	26	48	10	32	54	16	38	0	22	44	6	28	50
23	0	23	46	9	32	55	18	41	4	27	50	13	36	59	22	45	8	31	54	17	40	3	26	49	12	35	58	21	44	7	30	53	16	39	2	25
24	0	24	48	12	36	0	24	48	12	36	0	24	48	12	36	0	24	48	12	36	0	24	48	12	36	0	24	48	12	36	0	24	48	12	36	0
25	0	25	50	15	40	5	30	55	20	45	10	35	0	25	50	15	40	5	30	55	20	45	10	35	0	25	50	15	40	5	30	55	20	45	10	35
26	0	26	52	18	44	10	36	2	28	54	20	46	12	38	4	30	56	22	48	14	40	6	32	58	24	50	16	42	8	34	0	26	52	18	44	10
27	0	27	54	21	48	15	42	9	36	3	30	57	24	51	18	45	12	39	6	33	0	27	54	21	48	15	42	9	36	3	30	57	24	51	18	45
28	0	28	56	24	52	20	48	16	44	12	40	8	36	4	32	0	28	56	24	52	20	48	16	44	12	40	8	36	4	32	0	28	56	24	52	20
29	0	29	58	27	56	25	54	23	52	21	50	19	48	17	46	15	44	13	42	11	40	9	38	7	36	5	34	3	32	1	30	59	28	57	26	55
30	0	30	0	30	0	30	0	30	0	30	0	30	0	30	0	30	0	30	0	30	0	30	0	30	0	30	0	30	0	30	0	30	0	30	0	30
31	0	31	2	33	4	35	6	37	8	39	10	41	12	43	14	45	16	47	18	49	20	51	22	53	24	55	26	57	28	59	30	1	32	3	34	5
32	0	32	4	36	8	40	12	44	16	48	20	52	24	56	28	0	32	4	36	8	40	12	44	16	48	20	52	24	56	28	0	32	4	36	8	40
33	0	33	6	39	12	45	18	51	24	57	30	3	36	9	42	15	48	21	54	27	0	33	6	39	12	45	18	51	24	57	30	3	36	9	42	15
34	0	34	8	42	16	50	24	58	32	6	40	14	48	22	56	30	4	38	12	46	20	54	28	2	36	10	44	18	52	26	0	34	8	42	16	50
35	0	35	10	45	20	55	30	5	40	15	50	25	0	35	10	45	20	55	30	5	40	15	50	25	0	35	10	45	20	55	30	5	40	15	50	25
36	0	36	12	48	24	0	36	12	48	24	0	36	12	48	24	0	36	12	48	24	0	36	12	48	24	0	36	12	48	24	0	36	12	48	24	0
37	0	37	14	51	28	5	42	19	56	33	10	47	24	1	38	15	52	29	6	43	20	57	34	11	48	25	2	39	16	53	30	7	44	21	58	35
38	0	38	16	54	32	10	48	26	4	42	20	58	36	14	52	30	8	46	24	2	40	18	56	34	12	50	28	6	44	22	0	38	16	54	32	10
39	0	39	18	57	36	15	54	33	12	51	30	9	48	27	6	45	24	3	42	21	0	39	18	57	36	15	54	33	12	51	30	9	48	27	6	45
40	0	40	20	0	40	20	0	40	20	0	40	20	0	40	20	0	40	20	0	40	20	0	40	20	0	40	20	0	40	20	0	40	20	0	40	20
41	0	41	22	3	44	25	6	47	28	9	50	31	12	53	34	15	56	37	18	59	40	21	2	43	24	5	46	27	8	49	30	11	52	33	14	55
42	0	42	24	6	48	30	12	54	36	18	0	42	24	6	48	30	12	54	36	18	0	42	24	6	48	30	12	54	36	18	0	42	24	6	48	30
43	0	43	26	9	52	35	18	1	44	27	10	53	36	19	2	45	28	11	54	37	20	3	46	29	12	55	38	21	4	47	30	13	56	39	22	5
44	0	44	28	12	56	40	24	8	52	36	20	4	48	32	16	0	44	28	12	56	40	24	8	52	36	20	4	48	32	16	0	44	28	12	56	40
45	0	45	30	15	0	45	30	15	0	45	30	15	0	45	30	15	0	45	30	15	0	45	30	15	0	45	30	15	0	45	30	15	0	45	30	15
46	0	46	32	18	4	50	36	22	8	54	40	26	12	58	44	30	16	2	48	34	20	6	52	38	24	10	56	42	28	14	0	46	32	18	4	50
47	0	47	34	21	8	55	42	29	16	3	50	37	24	11	58	45	32	19	6	53	40	27	14	1	48	35	22	9	56	43	30	17	4	51	38	25
48	0	48	36	24	12	0	48	36	24	12	0	48	36	24	12	0	48	36	24	12	0	48	36	24	12	0	48	36	24	12	0	48	36	24	12	0
49	0	49	38	27	16	5	54	43	32	21	10	59	48	37	26	15	4	53	42	31	20	9	58	47	36	25	14	3	52	41	30	19	8	57	46	35
50	0	50	40	30	20	10	0	50	40	30	20	10	0	50	40	30	20	10	0	50	40	30	20	10	0	50	40	30	20	10	0	50	40	30	20	10
51	0	51	42	33	24	15	6	57	48	39	30	21	12	3	54	45	36	27	18	9	0	51	42	33	24	15	6	57	48	39	30	21	12	3	54	45
52	0	52	44	36	28	20	12	4	56	48	40	32	24	16	8	0	52	44	36	28	20	12	4	56	48	40	32	24	16	8	0	52	44	36	28	20
53	0	53	46	39	32	25	18	11	4	57	50	43	36	29	22	15	8	1	54	47	40	33	26	19	12	5	58	51	44	37	30	23	16	9	2	55
54	0	54	48	42	36	30	24	18	12	6	0	54	48	42	36	30	24	18	12	6	0	54	48	42	36	30	24	18	12	6	0	54	48	42	36	30
55	0	55	50	45	40	35	30	25	20	15	10	5	0	55	50	45	40	35	30	25	20	15	10	5	0	55	50	45	40	35	30	25	20	15	10	5
56	0	56	52	48	44	40	36	32	28	24	20	16	12	8	4	0	56	52	48	44	40	36	32	28	24	20	16	12	8	4	0	56	52	48	44	40
57	0	57	54	51	48	45	42	39	36	33	30	27	24	21	18	15	12	9	6	3	0	57	54	51	48	45	42	39	36	33	30	27	24	21	18	15
58	0	58	56	54	52	50	48	46	44	42	40	38	36	34	32	30	28	26	24	22	20	18	16	14	12	10	8	6	4	2	0	58	56	54	52	50
59	0	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25

Mod 60 Odd Numbers

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
3	0	3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51	54	57	0	3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51	54	57	0	3	6	9	12	15	18	21	24
5	0	5	10	15	20	25	30	35	40	45	50	55	0	5	10	15	20	25	30	35	40	45	50	55	0	5	10	15	20	25	30	35	40	45	50	55	0	5	10	15	20	25	30	35	40	45	50	55	0
7	0	7	14	21	28	35	42	49	56	3	10	17	24	31	38	45	52	59	6	13	20	27	34	41	48	55	2	9	16	23	30	37	44	51	58	5	12	19	26	33	40	47	54	1	8	15	22	29	36
9	0	9	18	27	36	45	54	3	12	21	30	39	48	57	6	15	24	33	42	51	0	9	18	27	36	45	54	3	12	21	30	39	48	57	6	15	24	33	42	51	0	9	18	27	36	45	54	3	12
11	0	11	22	33	44	55	6	17	28	39	50	1	12	23	34	45	56	7	18	29	40	51	2	13	24	35	46	57	8	19	30	41	52	3	14	25	36	47	58	9	20	31	42	53	4	15	26	37	48
13	0	13	26	39	52	5	18	31	44	57	10	23	36	49	2	15	28	41	54	7	20	33	46	59	12	25	38	51	4	17	30	43	56	9	22	35	48	1	14	27	40	53	6	19	32	45	58	11	24
15	0	15	30	45	0	15	30	45	0	15	30	45	0	15	30	45	0	15	30	45	0	15	30	45	0	15	30	45	0	15	30	45	0	15	30	45	0	15	30	45	0	15	30	45	0	15	30	45	0
17	0	17	34	51	8	25	42	59	16	33	50	7	24	41	58	15	32	49	6	23	40	57	14	31	48	5	22	39	56	13	30	47	4	21	38	55	12	29	46	3	20	37	54	11	28	45	2	19	36
19	0	19	38	57	16	35	54	13	32	51	10	29	48	7	26	45	4	23	42	1	20	39	58	17	36	55	14	33	52	11	30	49	8	27	46	5	24	43	2	21	40	59	18	37	56	15	34	53	12
21	0	21	42	3	24	45	6	27	48	9	30	51	12	33	54	15	36	57	18	39	0	21	42	3	24	45	6	27	48	9	30	51	12	33	54	15	36	57	18	39	0	21	42	3	24	45	6	27	48
23	0	23	46	9	32	55	18	41	4	27	50	13	36	59	22	45	8	31	54	17	40	3	26	49	12	35	58	21	44	7	30	53	16	39	2	25	48	11	34	57	20	43	6	29	52	15	38	1	24
25	0	25	50	15	40	5	30	55	20	45	10	35	0	25	50	15	40	5	30	55	20	45	10	35	0	25	50	15	40	5	30	55	20	45	10	35	0	25	50	15	40	5	30	55	20	45	10	35	0
27	0	27	54	21	48	15	42	9	36	3	30	57	24	51	18	45	12	39	6	33	0	27	54	21	48	15	42	9	36	3	30	57	24	51	18	45	12	39	6	33	0	27	54	21	48	15	42	9	36
29	0	29	58	27	56	25	54	23	52	21	50	19	48	17	46	15	44	13	42	11	40	9	38	7	36	5	34	3	32	1	30	59	28	57	26	55	24	53	22	51	20	49	18	47	16	45	14	43	12
31	0	31	2	33	4	35	6	37	8	39	10	41	12	43	14	45	16	47	18	49	20	51	22	53	24	55	26	57	28	59	30	1	32	3	34	5	36	7	38	9	40	11	42	13	44	15	46	17	48
33	0	33	6	39	12	45	18	51	24	57	30	3	36	9	42	15	48	21	54	27	0	33	6	39	12	45	18	51	24	57	30	3	36	9	42	15	48	21	54	27	0	33	6	39	12	45	18	51	24
35	0	35	10	45	20	55	30	5	40	15	50	25	0	35	10	45	20	55	30	5	40	15	50	25	0	35	10	45	20	55	30	5	40	15	50	25	0	35	10	45	20	55	30	5	40	15	50	25	0
37	0	37	14	51	28	5	42	19	56	33	10	47	24	1	38	15	52	29	6	43	20	57	34	11	48	25	2	39	16	53	30	7	44	21	58	35	12	49	26	3	40	17	54	31	8	45	22	59	36
39	0	39	18	57	36	15	54	33	12	51	30	9	48	27	6	45	24	3	42	21	0	39	18	57	36	15	54	33	12	51	30	9	48	27	6	45	24	3	42	21	0	39	18	57	36	15	54	33	12
41	0	41	22	3	44	25	6	47	28	9	50	31	12	53	34	15	56	37	18	59	40	21	2	43	24	5	46	27	8	49	30	11	52	33	14	55	36	17	58	39	20	1	42	23	4	45	26	7	48
43	0	43	26	9	52	35	18	1	44	27	10	53	36	19	2	45	28	11	54	37	20	3	46	29	12	55	38	21	4	47	30	13	56	39	22	5	48	31	14	57	40	23	6	49	32	15	58	41	24
45	0	45	30	15	0	45	30	15	0	45	30	15	0	45	30	15	0	45	30	15	0	45	30	15	0	45	30	15	0	45	30	15	0	45	30	15	0	45	30	15	0	45	30	15	0	45	30	15	0
47	0	47	34	21	8	55	42	29	16	3	50	37	24	11	58	45	32	19	6	53	40	27	14	1	48	35	22	9	56	43	30	17	4	51	38	25	12	59	46	33	20	7	54	41	28	15	2	49	36
49	0	49	38	27	16	5	54	43	32	21	10	59	48	37	26	15	4	53	42	31	20	9	58	47	36	25	14	3	52	41	30	19	8	57	46	35	24	13	2	51	40	29	18	7	56	45	34	23	12
51	0	51	42	33	24	15	6	57	48	39	30	21	12	3	54	45	36	27	18	9	0	51	42	33	24	15	6	57	48	39	30	21	12	3	54	45	36	27	18	9	0	51	42	33	24	15	6	57	48
53	0	53	46	39	32	25	18	11	4	57	50	43	36	29	22	15	8	1	54	47	40	33	26	19	12	5	58	51	44	37	30	23	16	9	2	55	48	41	34	27	20	13	6	59	52	45	38	31	24
55	0	55	50	45	40	35	30	25	20	15	10	5	0	55	50	45	40	35	30	25	20	15	10	5	0	55	50	45	40	35	30	25	20	15	10	5	0	55	50	45	40	35	30	25	20	15	10	5	0
57	0	57	54	51	48	45	42	39	36	33	30	27	24	21	18	15	12	9	6	3	0	57	54	51	48	45	42	39	36	33	30	27	24	21	18	15	12	9	6	3	0	57	54	51	48	45	42	39	36
59	0	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12

Mod 60

MOD 60																															
		Possible d Values																													
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
	1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
	7	0	7	14	21	28	35	42	49	56	3	10	17	24	31	38	45	52	59	6	13	20	27	34	41	48	55	2	9	16	23
	11	0	11	22	33	44	55	6	17	28	39	50	1	12	23	34	45	56	7	18	29	40	51	2	13	24	35	46	57	8	19
	13	0	13	26	39	52	5	18	31	44	57	10	23	36	49	2	15	28	41	54	7	20	33	46	59	12	25	38	51	4	17
	17	0	17	34	51	8	25	42	59	16	33	50	7	24	41	58	15	32	49	6	23	40	57	14	31	48	5	22	39	56	13
Possible	19	0	19	38	57	16	35	54	13	32	51	10	29	48	7	26	45	4	23	42	1	20	39	58	17	36	55	14	33	52	11
e	23	0	23	46	9	32	55	18	41	4	27	50	13	36	59	22	45	8	31	54	17	40	3	26	49	12	35	58	21	44	7
Values	29	0	29	58	27	56	25	54	23	52	21	50	19	48	17	46	15	44	13	42	11	40	9	38	7	36	5	34	3	32	1
	31	0	31	2	33	4	35	6	37	8	39	10	41	12	43	14	45	16	47	18	49	20	51	22	53	24	55	26	57	28	59
	37	0	37	14	51	28	5	42	19	56	33	10	47	24	1	38	15	52	29	6	43	20	57	34	11	48	25	2	39	16	53
	41	0	41	22	3	44	25	6	47	28	9	50	31	12	53	34	15	56	37	18	59	40	21	2	43	24	5	46	27	8	49
	43	0	43	26	9	52	35	18	1	44	27	10	53	36	19	2	45	28	11	54	37	20	3	46	29	12	55	38	21	4	47
	47	0	47	34	21	8	55	42	29	16	3	50	37	24	11	58	45	32	19	6	53	40	27	14	1	48	35	22	9	56	43
	49	0	49	38	27	16	5	54	43	32	21	10	59	48	37	26	15	4	53	42	31	20	9	58	47	36	25	14	3	52	41
	53	0	53	46	39	32	25	18	11	4	57	50	43	36	29	22	15	8	1	54	47	40	33	26	19	12	5	58	51	44	37
	59	0	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31
		Possible d Values																													
		30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
	1	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
	7	30	37	44	51	58	5	12	19	26	33	40	47	54	1	8	15	22	29	36	43	50	57	4	11	18	25	32	39	46	53
	11	30	41	52	3	14	25	36	47	58	9	20	31	42	53	4	15	26	37	48	59	10	21	32	43	54	5	16	27	38	49
	13	30	43	56	9	22	35	48	1	14	27	40	53	6	19	32	45	58	11	24	37	50	3	16	29	42	55	8	21	34	47
	17	30	47	4	21	38	55	12	29	46	3	20	37	54	11	28	45	2	19	36	53	10	27	44	1	18	35	52	9	26	43
Possible	19	30	49	8	27	46	5	24	43	2	21	40	59	18	37	56	15	34	53	12	31	50	9	28	47	6	25	44	3	22	41
e	23	30	53	16	39	2	25	48	11	34	57	20	43	6	29	52	15	38	1	24	47	10	33	56	19	42	5	28	51	14	37
Values	29	30	59	28	57	26	55	24	53	22	51	20	49	18	47	16	45	14	43	12	41	10	39	8	37	6	35	4	33	2	31
	31	30	1	32	3	34	5	36	7	38	9	40	11	42	13	44	15	46	17	48	19	50	21	52	23	54	25	56	27	58	29
	37	30	7	44	21	58	35	12	49	26	3	40	17	54	31	8	45	22	59	36	13	50	27	4	41	18	55	32	9	46	23
	41	30	11	52	33	14	55	36	17	58	39	20	1	42	23	4	45	26	7	48	29	10	51	32	13	54	35	16	57	38	19
	43	30	13	56	39	22	5	48	31	14	57	40	23	6	49	32	15	58	41	24	7	50	33	16	59	42	25	8	51	34	17
	47	30	17	4	51	38	25	12	59	46	33	20	7	54	41	28	15	2	49	36	23	10	57	44	31	18	5	52	39	26	13
	49	30	19	8	57	46	35	24	13	2	51	40	29	18	7	56	45	34	23	12	1	50	39	28	17	6	55	44	33	22	11
	53	30	23	16	9	2	55	48	41	34	27	20	13	6	59	52	45	38	31	24	17	10	3	56	49	42	35	28	21	14	7
	59	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Mod 60

		Possible d Values																													
		30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
	1	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
	7	30	37	44	51	58	5	12	19	26	33	40	47	54	1	8	15	22	29	36	43	50	57	4	11	18	25	32	39	46	53
	11	30	41	52	3	14	25	36	47	58	9	20	31	42	53	4	15	26	37	48	59	10	21	32	43	54	5	16	27	38	49
	13	30	43	56	9	22	35	48	1	14	27	40	53	6	19	32	45	58	11	24	37	50	3	16	29	42	55	8	21	34	47
	17	30	47	4	21	38	55	12	29	46	3	20	37	54	11	28	45	2	19	36	53	10	27	44	1	18	35	52	9	26	43
Possible	19	30	49	8	27	46	5	24	43	2	21	40	59	18	37	56	15	34	53	12	31	50	9	28	47	6	25	44	3	22	41
e	23	30	53	16	39	2	25	48	11	34	57	20	43	6	29	52	15	38	1	24	47	10	33	56	19	42	5	28	51	14	37
Values	29	30	59	28	57	26	55	24	53	22	51	20	49	18	47	16	45	14	43	12	41	10	39	8	37	6	35	4	33	2	31
	31	30	1	32	3	34	5	36	7	38	9	40	11	42	13	44	15	46	17	48	19	50	21	52	23	54	25	56	27	58	29
	37	30	7	44	21	58	35	12	49	26	3	40	17	54	31	8	45	22	59	36	13	50	27	4	41	18	55	32	9	46	23
	41	30	11	52	33	14	55	36	17	58	39	20	1	42	23	4	45	26	7	48	29	10	51	32	13	54	35	16	57	38	19
	43	30	13	56	39	22	5	48	31	14	57	40	23	6	49	32	15	58	41	24	7	50	33	16	59	42	25	8	51	34	17
	47	30	17	4	51	38	25	12	59	46	33	20	7	54	41	28	15	2	49	36	23	10	57	44	31	18	5	52	39	26	13
	49	30	19	8	57	46	35	24	13	2	51	40	29	18	7	56	45	34	23	12	1	50	39	28	17	6	55	44	33	22	11
	53	30	23	16	9	2	55	48	41	34	27	20	13	6	59	52	45	38	31	24	17	10	3	56	49	42	35	28	21	14	7
	59	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Mod 60

		Possible d Values																													
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
	1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
	7	0	7	14	21	28	35	42	49	56	3	10	17	24	31	38	45	52	59	6	13	20	27	34	41	48	55	2	9	16	23
	11	0	11	22	33	44	55	6	17	28	39	50	1	12	23	34	45	56	7	18	29	40	51	2	13	24	35	46	57	8	19
	13	0	13	26	39	52	5	18	31	44	57	10	23	36	49	2	15	28	41	54	7	20	33	46	59	12	25	38	51	4	17
	17	0	17	34	51	8	25	42	59	16	33	50	7	24	41	58	15	32	49	6	23	40	57	14	31	48	5	22	39	56	13
Possible	19	0	19	38	57	16	35	54	13	32	51	10	29	48	7	26	45	4	23	42	1	20	39	58	17	36	55	14	33	52	11
e	23	0	23	46	9	32	55	18	41	4	27	50	13	36	59	22	45	8	31	54	17	40	3	26	49	12	35	58	21	44	7
Values	29	0	29	58	27	56	25	54	23	52	21	50	19	48	17	46	15	44	13	42	11	40	9	38	7	36	5	34	3	32	1
	31	0	31	2	33	4	35	6	37	8	39	10	41	12	43	14	45	16	47	18	49	20	51	22	53	24	55	26	57	28	59
	37	0	37	14	51	28	5	42	19	56	33	10	47	24	1	38	15	52	29	6	43	20	57	34	11	48	25	2	39	16	53
	41	0	41	22	3	44	25	6	47	28	9	50	31	12	53	34	15	56	37	18	59	40	21	2	43	24	5	46	27	8	49
	43	0	43	26	9	52	35	18	1	44	27	10	53	36	19	2	45	28	11	54	37	20	3	46	29	12	55	38	21	4	47
	47	0	47	34	21	8	55	42	29	16	3	50	37	24	11	58	45	32	19	6	53	40	27	14	1	48	35	22	9	56	43
	49	0	49	38	27	16	5	54	43	32	21	10	59	48	37	26	15	4	53	42	31	20	9	58	47	36	25	14	3	52	41
	53	0	53	46	39	32	25	18	11	4	57	50	43	36	29	22	15	8	1	54	47	40	33	26	19	12	5	58	51	44	37
	59	0	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31

Example

From the Mod 60 tables we see that we could choose the following for e and d:

e	d
1	1
7	43
11	11
13	37
17	53
19	19
23	47
29	29
31	31
37	13
41	41
43	7
47	23
49	49
53	17
59	59

Not a good choice!

We choose:

e = 7, d = 43

Example

Suppose we want to encrypt a message containing the number 9

Let m = 9

The encryption algorithm gives:

$$c = m^e \bmod n$$

$$= 9^7 \bmod 77$$

$$= 4782969 \bmod 77$$

$$= 37 + 4782932 \bmod 77$$

$$= 37 + 62116 \times 77 \bmod 77$$

$$= 37$$

We send the ciphertext value 37

Example

Decryption would be performed as follows:

c = 37 and $K = (43, 77)$

The decryption algorithm gives:

$$m = c^d \bmod n$$

$$= 37^{43} \bmod 77$$

$$= ??????$$

This is a large number to calculate and leads to overflow errors.

We therefore need an alternative approach to evaluate the above

Example

Again we turn to a property from Number Theory to help us out:

$$ab \bmod n = (a \bmod n \times b \bmod n) \bmod n$$

This allows us to work with and simplify smaller parts of the calculation a bit at a time thus avoiding overflow errors

Note the use of powers of 2 in the following calculation

Example

$$
\begin{aligned}
m &= c^d \bmod n \\
&= 37^{43} \bmod 77 \\
&= 37^{32} \times 37^8 \times 37^2 \times 37^1 \bmod 77 \\
&= (37^8)^4 \times 37^8 \times 37^2 \times 37^1 \bmod 77 \\
&= ((37^2)^4)^4 \times (37^2)^4 \times 37^2 \times 37^1 \bmod 77 \\
&= (60^4)^4 \times 60^4 \times 60 \times 37 \bmod 77 \\
&= 53^4 \times 53 \times 60 \times 37 \bmod 77 \\
&= 60 \times 53 \times 60 \times 37 \bmod 77 \\
&= 7059600 \bmod 77 \\
&= 9 + 7059591 \bmod 77 \\
&= 9 + 91683 \times 77 \bmod 77 \\
&= 9
\end{aligned}
$$

Hence the original message is recovered.