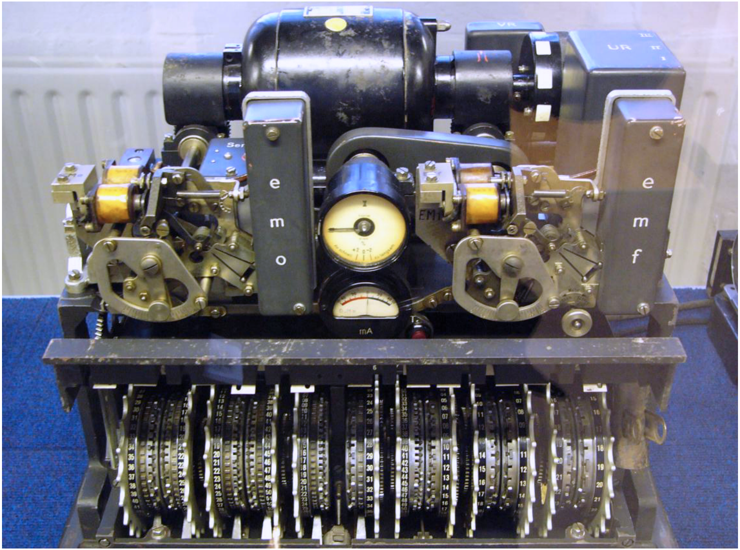
CS4236 Cryptography Theory and Practice Topic 9 - SPNs and PK math background

Hugh Anderson

National University of Singapore
School of Computing

October, 2022

The Lorenz cipher machine...



Outline

1. **SPNs in symmetric systems**
   - A toy SPN
   - DES
   - AES
2. **Cryptanalysis of SPNs**
   - Differential cryptanalysis
   - Linear cryptanalysis
   - Integral cryptanalysis
3. **More math foundations for PK**
   - The landscape of math for crypto
   - Algorithms, with complexity
   - Problems in $\mathbb{Z}_p$ and $\mathbb{Z}_N$

The story so far ... where are we?

The last 8 weeks: weekly steps we have taken

1. We had a historical/contextual introduction in Session 1.
2. We progressed from perfect *secrecy* to perfect *indistinguishability*.
3. *Perfectly* indistinguishable was relaxed to give *computationally* indistinguishable. An EAV-Secure system was constructed with a PRG.
4. The notion of CPA-secure was developed, and achieved with a PRF.
5. Modes, the "padding oracle", and CCA-Security were outlined.
6. We looked at cryptographic MACs and *authenticated* encryption.
7. We began looking at hashes, with definitions, games, and applications.
8. Last week we continued with hash applications, primarily commitment. We then looked at attack scenarios for hashes, the structure and use of the lovely rainbow tables, and also attacks exploiting the "birthday" not-a-paradox. Floyd's algorithm allows for collision detection with only $\mathcal{O}(1)$ memory usage.

SPN

Substitution/Permutation Networks

Many block ciphers such as DES and AES are variants of SPN: Substitution and Permutation Networks.

The general idea is that there are many *rounds* of substitution and permutation performed.

SPN ciphers are easy to understand

The ciphers achieve confusion and diffusion with two main components.

1. $S(x)$: the S-box (substitution box); and
2. $P(x)$: the permutation.

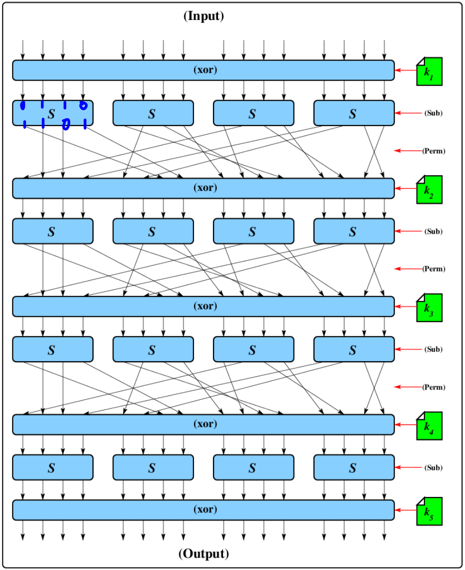
Both $S(x)$ and $P(x)$ are fixed: they are part of the algorithms. Hence, the definitions of both $S(x)$ and $P(x)$ are not secret. In contrast to this, the “classical” substitution cipher treats the substitution table itself as the secret key.

The secret key generates the subkeys, which are “xor-ed” with the output at every round. The encryption is shown with a “toy” SPN in the next slide.

The “toy” SPN from the textbook

x	$S(x)$)
0	0
1	b
2	5
3	1
4	6
5	8
6	d
7	4
8	f
9	7
a	2
b	c
c	9
d	3
e	e
f	a

z	$P(z)$
1	7
2	2
3	3
4	8
5	12
6	5
7	11
8	9
9	10
10	1
11	14
12	13
13	4
14	6
15	16
16	15



SPN

Substitution/Permutation Networks

The S-boxes are fixed, mostly for historical/hardware reasons. Early implementations in hardware dictated many design decisions. DES and AES both use fixed S-boxes.

Some questions for discussion on the toy SPN

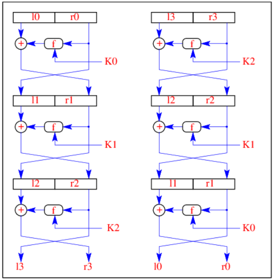
1. What if the permutation is the identity function (i.e. $P(z) = z$ for all z)? Give a chosen plaintext attack.
2. What if the S-box is the identity function, i.e. $S(x) = x$?
3. What if the S-box is the shift cipher, $S(x) = (x + 1) \bmod 16$?

DES - Data Encryption Standard

DES - a (hardware oriented) BLOCK cipher

DES was first proposed by IBM (1974) using 128 bit keys. The Security was reduced by NSA (the National Security Agency) to a 56 bit key. The (shared) 56 bit key generates 16 48-bit subkeys, which each control a *round*.

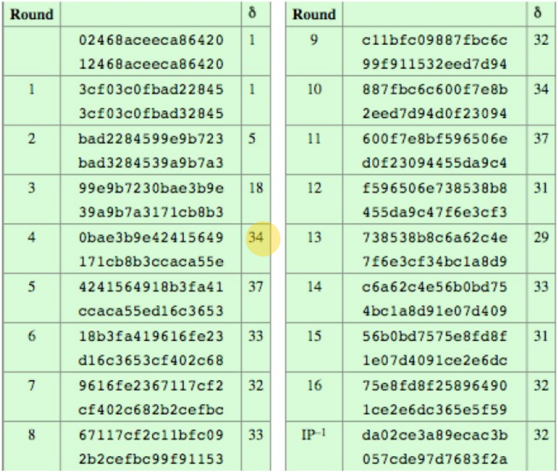
At the core of DES is the Feistel network



Each of the 16 stages (rounds) of DES uses a Feistel structure which encrypts a 64 bit value into another 64 bit value using the 48 bit subkey. There is a substitution on the left data half, based on a round function. Then we have a permutation - swapping halves.

DES avalanche - 1 bit change in plaintext

Number of bits changed/64 on each round



Round		δ
	02468aceeca86420 12468aceeca86420	1
1	3cf03c0fbad22845 3cf03c0fbad32845	1
2	bad2284599e9b723 bad3284539a9b7a3	5
3	99e9b7230bae3b9e 39a9b7a3171cb8b3	18
4	0bae3b9e42415649 171cb8b3ccaca55e	34
5	4241564918b3fa41 ccaca55ed16c3653	37
6	18b3fa419616fe23 d16c3653cf402c68	33
7	9616fe2367117cf2 cf402c682b2cefbc	32
8	67117cf2c11bfc09 2b2cefbc99f91153	33
9	c11bfc09887fbc6c 99f911532eed7d94	32
10	887fbc6c600f7e8b 2eed7d94d0f23094	34
11	600f7e8bf596506e d0f23094455da9c4	37
12	f596506e738538b8 455da9c47f6e3cf3	31
13	738538b8c6a62c4e 7f6e3cf34bc1a8d9	29
14	c6a62c4e56b0bd75 4bc1a8d91e07d409	33
15	56b0bd7575e8fd8f 1e07d4091ce2e6dc	31
16	75e8fd8f25896490 1ce2e6dc365e5f59	32
IP^{-1}	da02ce3a89ecac3b 057cde97d7683f2a	32

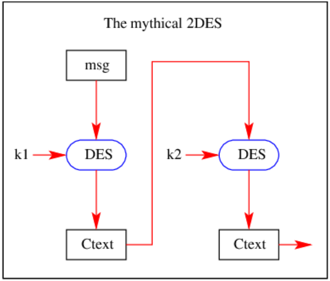
DES considered harmful

Or at least... weak ...

56-bit keys have $2^{56} = 7.2 \times 10^{16}$ values. A brute force search looks like it might be hard, but consider the speedup in computers from the 1970s onwards. In 1997 using a network of computers on the Internet, it took a few months. In 1998, on dedicated h/w (EFF), it took a few days

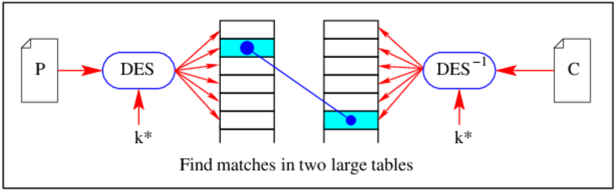
http://en.wikipedia.org/wiki/EFF_DES_cracker

In 1999, 22hrs (and so on). How long does it take now? We must consider alternatives to DES: 2DES?



The problem with 2DES

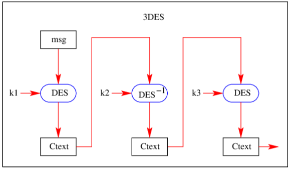
2DES meet-in-the-middle attack



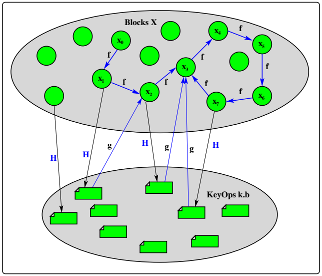
Find matches in two large tables

Starts with the attacker having a plaintext, ciphertext pair $\langle p, c\rangle$. The attacker computes two tables: $E(k, p)$ for each of the 2^{56} keys, and $D(k, c)$ for each of the 2^{56} keys. For each match in the two tables, you have found a possible key, with only 2^{57} DES operations (ie - not 2^{112}).

What we use: 3DES. Attack needs 2^{112} operations.



2DES memory bound? Consider Floyd again



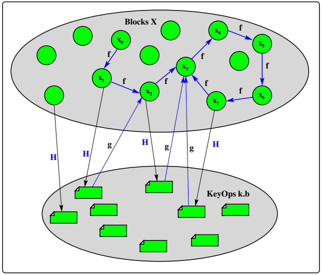
Apply Floyd by using $f(x) \stackrel{\text{def}}{=} g(\mathcal{H}(x))$

Consider the worlds of possible intermediate blocks $x \in X$, and keyop elements $k \cdot b$: a (random) 56-bit key k, and a (random) bit b selecting encryption of a plaintext p or decryption of a matching ciphertext c:

$$g(k \cdot b) \quad = \quad \begin{cases} b = 1 & \rightarrow \quad \mathrm{Enc}_k(p) \\ b = 0 & \rightarrow \quad \mathrm{Dec}_k(c) \end{cases}$$

If Floyd finds a collision, then with probability 0.5, one is an encryption and one a decryption, and we have candidate keys for 2DES.

Code changes for the collision attack



Few code changes...

```
def g(x):
   k,b = keyop(x)
   encryptor = DES.new(binascii.unhexlify(k)[:8], DES.MODE_ECB)
   if b:
      return encryptor.encrypt(plaintext)
   else:
      return encryptor.decrypt(ciphertext)

def H(x):
    return md5(x).hexdigest()

def f(x):
    return g(H(x))
```

AES intro

Due to rising worries about DES

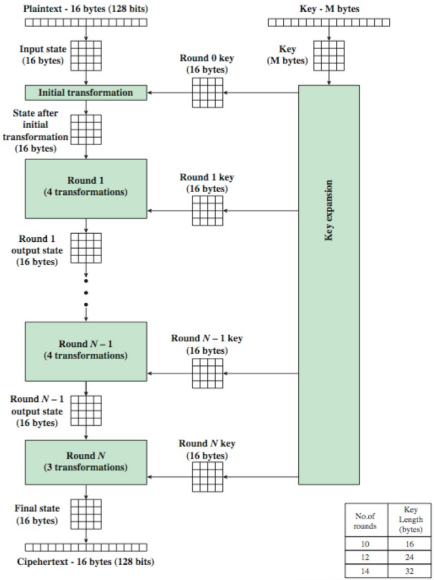
In 1997 NIST called for proposals for a royalty-free and public symmetric block cipher. In 1998 and 1999 the finalists were evaluated in public (and of course by NSA). The winner (Rijndael) became the FIPS 197 (Federal Information Processing Standard 197), commonly known as AES. The standard comprises three ciphers selected from Rijndael.
The authors were two Belgians, Dr Joan Daemen and Dr Vincent Rijmen (Rijndael is a play on their names).

The US Advanced Encryption Standard

The standard comprises three ciphers selected from Rijndael (AES-128, AES-192 and AES-256 with key sizes of 128, 192 and 256 bits). It is a symmetric block-based data encryption standard with a 128-bit block size. Designed to be resistant against known attacks, have speed and code compactness on many CPUs, along with design simplicity (so it can weather criticism).
It is not a feistel structure, instead it is state-based. The algorithm is normally specified in code form. Uses state, rounds, substitution, shifts, mixing, and roundkeys...

AES operations

10/12/14 rounds


Plaintext - 16 bytes (128 bits)
Key - M bytes
Input state (16 bytes)
Key (M bytes)
Round 0 key (16 bytes)
Initial transformation
State after initial transformation (16 bytes)
Round 1 (4 transformations)
Round 1 key (16 bytes)
Key expansion
Round 1 output state (16 bytes)
Round N – 1 (4 transformations)
Round N – 1 key (16 bytes)
Round N – 1 output state (16 bytes)
Round N (3 transformations)
Round N key (16 bytes)
Final state (16 bytes)
Cipehertext - 16 bytes (128 bits)

No.of rounds	Key Length (bytes)
10	16
12	24
14	32

AES design

Transformations...

The main data structure is a data block of 4 columns of 4 bytes, and is called the state. The key is expanded to an array of (32-bit) words.
There is an initial XOR with the key (AddRoundKey), and then the number of rounds is dependent on the key size (10,12,14). Each round modifies the state:

byte substitution: *one S-box used on every byte*
shift rows: *permute bytes between groups/columns*
mix columns: *substitutions using matrix multiply of groups*
add round key: *XOR state with key material*

There is an incomplete last round (No MixColumns operation). All of the rounds can be done efficiently, using fast XOR operations, and table lookup.

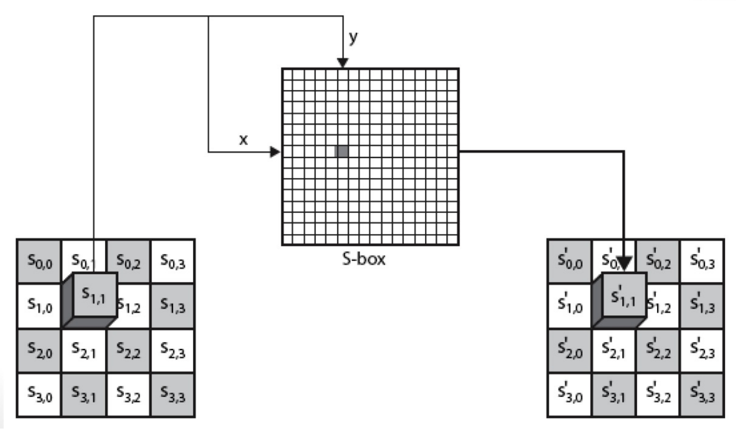
AES algorithm

AES code

```
Cipher(byte in[4*4],byte out[4*4],word w[4*(Nr+1)])
begin
byte state[4,4]
  state = in
  AddRoundKey(state, w[0, 3])
  for round = 1 step 1 to Nr-1
    SubBytes(state)
    ShiftRows(state)
    MixColumns(state)
    AddRoundKey(state, w[round*4, (round+1)*4-1])
  end for
  SubBytes(state)
  ShiftRows(state)
  AddRoundKey(state, w[Nr*4, (Nr+1)*4-1])
  out = state
end
```

AES operations...

Substitute bytes



y
x
S-box
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

GF(2^8) and AES

The byte substitution S-Box in AES:

The AES S-Box is built on polynomial computations done in GF(2^8), using the irreducible polynomial $x^8 + x^4 + x^3 + x + 1 = 100011011$.

The first step is to calculate the multiplicative inverse of the (non-zero) byte value in the field, giving an 8-bit value $\left[x^7 \ldots x^0\right]$.

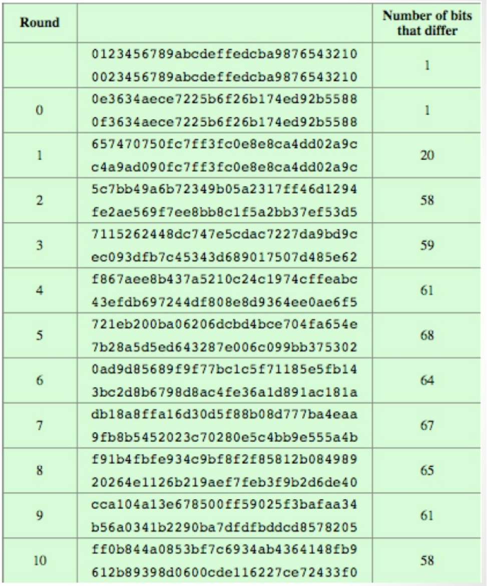
The substitution is then

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x^0 \\ x^1 \\ x^2 \\ x^3 \\ x^4 \\ x^5 \\ x^6 \\ x^7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

The tables for all the byte values can be pre-computed.

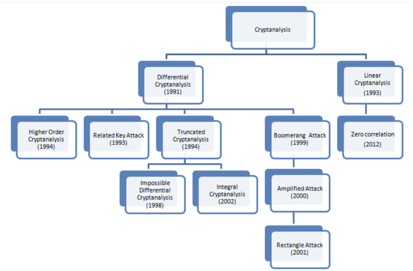
AES avalanche - 1 bit change in plaintext

Number of bits changed/128 on each round



Round		Number of bits that differ
	0123456789abcdeffedcba9876543210 0023456789abcdeffedcba9876543210	1
0	0e3634aece7225b6f26b174ed92b5588 0f3634aece7225b6f26b174ed92b5588	1
1	657470750fc7ff3fc0e8e8ca4dd02a9c c4a9ad090fc7ff3fc0e8e8ca4dd02a9c	20
2	5c7bb49a6b72349b05a2317ff46d1294 fe2ae569f7ee8bb8c1f5a2bb37ef53d5	58
3	7115262448dc747e5cdac7227da9bd9c ec093dfb7c45343d689017507d485e62	59
4	f867aee8b437a5210c24c1974cffeabc 43efdb697244df808e8d9364ee0ae6f5	61
5	721eb200ba06206dcbd4bce704fa654e 7b28a5d5ed643287e006c099bb375302	68
6	0ad9d85689f9f77bc1c5f71185e5fb14 3bc2d8b6798d8ac4fe36a1d891ac181a	64
7	db18a8ffa16d30d5f88b08d777ba4eaa 9fb8b5452023c70280e5c4bb9e555a4b	67
8	f91b4fbfe934c9bf8f2f85812b084989 20264e1126b219aef7feb3f9b2d6de40	65
9	cca104a13e678500ff59025f3bafaa34 b56a0341b2290ba7dfdfbddcd8578205	61
10	ff0b844a0853bf7c6934ab4364148fb9 612b89398d0600cde116227ce72433f0	58

Differential cryptanalysis



Cryptanalysis

Differential Cryptanalysis (1991)

Linear Cryptanalysis (1993)

Higher Order Cryptanalysis (1994)

Related Key Attack (1993)

Truncated Cryptanalysis (1994)

Boomerang Attack (1999)

Zero correlation (2012)

Impossible Differential Cryptanalysis (1998)

Integral Cryptanalysis (2002)

Amplified Attack (2000)

Rectangle Attack (2001)

A chosen plaintext attack (pairs of plaintexts)

The attacker (by analysis of the S-boxes and permutations) determines which input bits affect which output bits with a higher likelihood. Then the attacker queries the system with chosen plaintexts, with pairs of messages that flip the bits of interest. By measuring the differences in the outputs, we can start identifying keys... example to follow.

Such an attacker capability is realistic - for example, a smartcard/SIM card may be able to act as an encryption oracle, that is, on input of a plaintext m, it will output the ciphertext encrypted using the key k.

What *exactly* are differentials?

x	$S(x))$	$x \oplus 1111$	$S(x \oplus 1111)$	$S(x) \oplus S(x \oplus 1111)$
0000	0000	1111	1010	1010
0001	1011	1110	1110	0101
0010	0101	1101	0011	0110
0011	0001	1100	1001	1000
0100	0110	1011	1100	1010
0101	1000	1010	0010	1010
0110	1101	1001	0111	1010
0111	0100	1000	1111	1011
1000	1111	0111	0100	1011
1001	0111	0110	1101	1010
1010	0010	0101	1000	1010
1011	1100	0100	0110	1010
1100	1001	0011	0001	1000
1101	0011	0010	0101	0110
1110	1110	0001	1011	0101
1111	1010	0000	0000	1010

Example above is for an input bit difference of 1111

In this example, we see that in $\frac{8}{16}$ cases, two of four output bits change ($b_3, b_1 = 1010$) and $\Pr[\langle \Delta_x, \Delta_y \rangle] = \Pr[\langle 1111, 1010 \rangle] = \frac{1}{2}$.

Tabulation of *all* differentials

	0	1	2	3	4	5	6	7	8	9	*a*	*b*	*c*	*d*	*e*	*f*
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	4	0	0	0	2	2	2	2	0	0	4	0
2	0	0	0	0	0	2	0	2	0	2	2	4	2	2	0	0
3	0	2	2	4	0	4	0	0	0	0	0	0	0	2	2	0
4	0	0	0	2	2	2	6	0	2	0	0	0	2	0	0	0
5	0	2	2	0	0	0	0	0	4	0	0	0	4	2	2	0
6	0	2	0	2	0	0	0	0	0	2	0	2	0	4	0	4
7	0	2	0	0	2	4	2	2	0	2	0	0	0	2	0	0
8	0	0	0	2	0	0	0	2	0	0	0	2	2	2	2	4
9	0	2	0	2	2	2	0	4	0	2	2	0	0	0	0	0
a	0	0	4	0	2	0	2	4	2	0	2	0	0	0	0	0
b	0	0	2	0	0	0	2	0	0	2	0	0	4	2	4	0
c	0	0	0	0	0	0	0	0	4	4	0	4	0	0	0	4
d	0	4	2	2	0	0	2	2	0	0	0	0	0	0	0	4
e	0	2	4	2	4	0	0	0	0	0	0	0	2	0	2	0
f	0	0	0	0	0	2	2	0	2	0	8	2	0	0	0	0

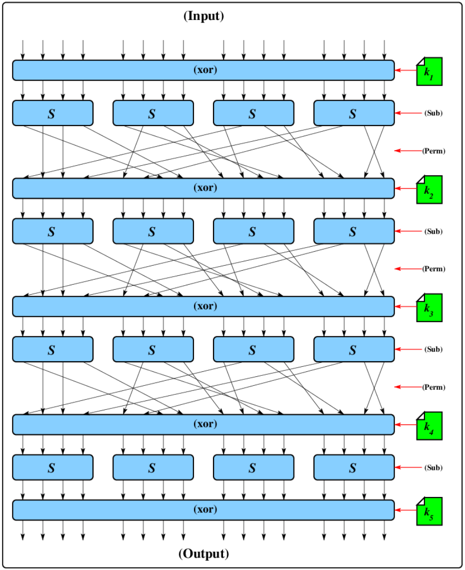
Differentials for all possible inputs in the toy SPN

By tradition, the table is given as a number (in this case out of 16), not a probability. We can see the highlighted element from before.

SPN again

x	$S(x)$
0	0
1	b
2	5
3	1
4	6
5	8
6	d
7	4
8	f
9	7
a	2
b	c
c	9
d	3
e	e
f	a

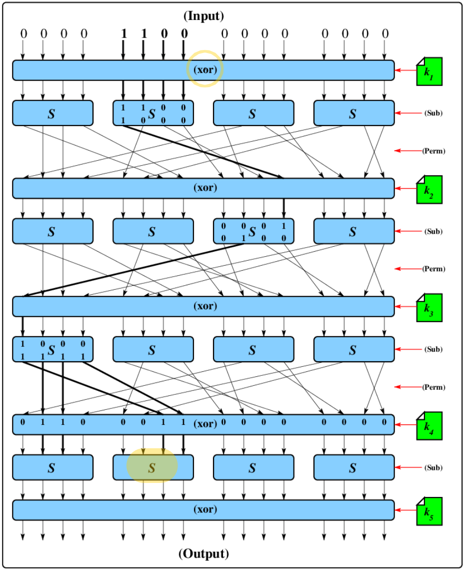
z	$P(z)$
1	7
2	2
3	3
4	8
5	12
6	5
7	11
8	9
9	10
10	1
11	14
12	13
13	4
14	6
15	16
16	15



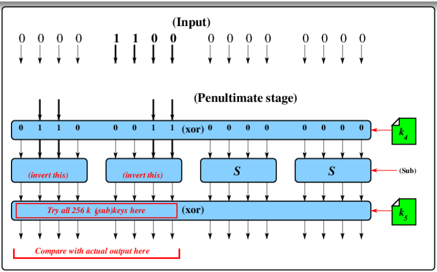
SPN again with $\Delta_x = 0000110000000000$ highlighted

x	$S(x))$
0	0
1	*b*
2	5
3	1
4	6
5	8
6	*d*
7	4
8	*f*
9	7
a	2
b	*c*
c	9
d	3
e	*e*
f	*a*

z	$P(z)$
1	7
2	2
3	3
4	8
5	12
6	5
7	11
8	9
9	10
10	1
11	14
12	13
13	4
14	6
15	16
16	15



Input bits affecting last stage output bits



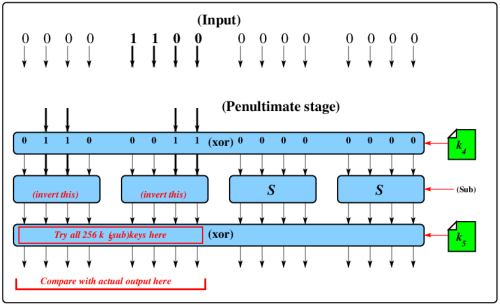
All bits affect all other bits, but...

The highlighted bits are *disproportionally* affected. If the 16 input bits were affecting output bits more or less at random, then $\Pr[\langle \Delta_{\text{in}}, \Delta_{\text{out}} \rangle]$ should be about $2^{-16} = \frac{1}{65536}$. But by following each S-Box in the chain, we have

$$\Pr[\langle \Delta_{\text{in}}, \Delta_{\text{out}} \rangle] = \Pr[\langle 0000110000000000, 0110001100000000 \rangle] = \frac{1}{64}$$

We use a CPA, and measure the differential probabilities for the 256 different values of the 8 most significant bits of k_5. The correct value of the key should give the expected differential 01100011 occuring with probabilty $\frac{1}{64}$, not $\frac{1}{256}$.

More on the algorithm

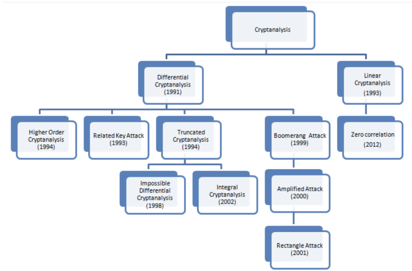


So far we only have retrieved 8 bits of k_5

The procedure is repeated using another differential chain, to retrieve the 8 other bits of the round key k_5. Once all of k_5 is known, the process can be repeated for k_4 and so on until all round keys are retrieved.

The attack was first described by Biham and Shamir, and working attacks on DES were demonstrated in 1993. New proposed ciphers are always analysed to see if they are susceptible to differential cryptanalysis.

Linear cryptanalysis



Cryptanalysis

Differential Cryptanalysis (1991)

Linear Cryptanalysis (1993)

Higher Order Cryptanalysis (1994)

Related Key Attack (1993)

Truncated Cryptanalysis (1994)

Boomerang Attack (1999)

Zero correlation (2012)

Impossible Differential Cryptanalysis (1998)

Integral Cryptanalysis (2002)

Amplified Attack (2000)

Rectangle Attack (2001)

A known (i.e. not chosen) plaintext attack

In linear cryptanalysis, the attacker considers *linear* relationships between input, output and keys. Again, the attacker must have access to a large number of plaintexts and corresponding ciphertexts.

Such an attacker capability is also realistic. Again, some smartcard protocols can be observed while they are running (although perhaps not modified - i.e. not a CPA).

In order to understand the linear cryptanalytic attack, we need to understand some preliminary probabilistic ideas used in the attack.

Preliminaries for linear cryptanalysis

Definition of biased variables

If X is a random variable that takes values from $\{0, 1\}$, then the *bias* of X is

$$\varepsilon(X) = \Pr[X = 0] - \frac{1}{2}$$

Consider two independent variables X_1 and X_2 . Suppose the bias of X_1 is 0.1 and bias of X_2 is -0.2, then what is the bias of $X_1 \oplus X_2$?

$$\begin{aligned}
\Pr[X_1 \oplus X_2 = 0] &= \Pr[(X_1 = 0 \wedge X_2 = 0) \vee (X_1 = 1 \wedge X_2 = 1)] \\
&= \Pr[X_1 = 0 \wedge X_2 = 0] + \Pr[X_1 = 1 \wedge X_2 = 1] \\
&= \Pr[X_1 = 0] \times \Pr[X_2 = 0] + \Pr[X_1 = 1] \times \Pr[X_2 = 1] \\
&= 0.6 \times 0.3 + 0.4 \times 0.7 \\
&= 0.46
\end{aligned}$$

So the bias is $\varepsilon(X_1 \oplus X_2) = -0.04$.

We see that XOR-ing independent binary variables reduces the bias (or at least does not increase it)

Preliminaries for linear cryptanalysis

The "piling-up" lemma

The piling-up lemma gives a formula for the bias for the random variable $X_1 \oplus X_2 \oplus \ldots$. The *bias* of $X = X_1 \oplus X_2 \oplus \ldots \oplus X_k$ is

$$\varepsilon(X) = 2^{k-1} \times \varepsilon_1 \times \varepsilon_1 \times \ldots \times \varepsilon_k$$

Note that the random variables have to be independent. Suppose that $Y = (1 - X)$, and the bias of X is 0.2. Hence, the bias of Y is -0.2. What is the bias of $X \oplus Y$?

The lemma implies that XOR-ing independent binary variables always reduces the bias (or at least does not increase it). Moreover, the output is unbiased if and only if there is at least one unbiased input variable.

We use bias to "approximate" SPNs

We investigate the S-box, calculating the bias of xor'ing each of its input-output combinations. This leads to the notion of a *linear approximation* of the S-box.

4 + 4 “bit” variables for the S-box

x	$S(x))$
0	0
1	b
2	5
3	1
4	6
5	8
6	d
7	4
8	f
9	7
a	2
b	c
c	9
d	3
e	e
f	a

$\longrightarrow$

X_3	X_2	X_1	X_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	1	0	1	1
0	0	1	0	0	1	0	1
0	0	1	1	0	0	0	1
0	1	0	0	0	1	1	0
0	1	0	1	1	0	0	0
0	1	1	0	1	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	1	1
1	0	0	1	0	1	1	1
1	0	1	0	0	0	1	0
1	0	1	1	1	1	0	0
1	1	0	0	1	0	0	1
1	1	0	1	0	0	1	1
1	1	1	0	1	1	1	0
1	1	1	1	1	0	1	0

Bit variables from the input and output bits of the S-Box

Instead of thinking of the 4-bit input and 4-bit output of the S-Box, we imagine 8 boolean variables.

Bias of $Z_{1,7} = \varepsilon(X_0 \oplus Y_2 \oplus Y_1 \oplus Y_0) = +\frac{1}{8}$

X_0	$\oplus$	Y_2	$\oplus$	Y_1	$\oplus$	Y_0	$\longrightarrow$	$Z_{1,7}$
0	$\oplus$	0	$\oplus$	0	$\oplus$	0	$\longrightarrow$	0
1	$\oplus$	0	$\oplus$	1	$\oplus$	1	$\longrightarrow$	1
0	$\oplus$	1	$\oplus$	0	$\oplus$	1	$\longrightarrow$	0
1	$\oplus$	0	$\oplus$	0	$\oplus$	1	$\longrightarrow$	0
0	$\oplus$	1	$\oplus$	1	$\oplus$	0	$\longrightarrow$	0
1	$\oplus$	0	$\oplus$	0	$\oplus$	0	$\longrightarrow$	1
0	$\oplus$	1	$\oplus$	0	$\oplus$	1	$\longrightarrow$	0
1	$\oplus$	1	$\oplus$	0	$\oplus$	0	$\longrightarrow$	0
0	$\oplus$	1	$\oplus$	1	$\oplus$	1	$\longrightarrow$	1
1	$\oplus$	1	$\oplus$	1	$\oplus$	1	$\longrightarrow$	0
0	$\oplus$	0	$\oplus$	1	$\oplus$	0	$\longrightarrow$	1
1	$\oplus$	1	$\oplus$	0	$\oplus$	0	$\longrightarrow$	0
0	$\oplus$	0	$\oplus$	0	$\oplus$	1	$\longrightarrow$	1
1	$\oplus$	0	$\oplus$	1	$\oplus$	1	$\longrightarrow$	1
0	$\oplus$	1	$\oplus$	1	$\oplus$	0	$\longrightarrow$	0
1	$\oplus$	0	$\oplus$	1	$\oplus$	0	$\longrightarrow$	0

Worked example of one input bit and 3 output bits

In this example we see how we can determine $Z_{1,7} = X_0 \oplus Y_2 \oplus Y_1 \oplus Y_0$ for all input values. There is a positive bias $\varepsilon(Z_{1,7}) = +\frac{1}{8}$. More 0s than 1s!

The bias of selection combinations

We can consider various possible $\oplus$ combinations

For example:

$$\begin{aligned} Z_{1,7} &= X_0 \oplus Y_2 \oplus Y_1 \oplus Y_0 \\ \text{or} \qquad & \\ Z_{b,4} &= X_3 \oplus X_1 \oplus X_0 \oplus Y_2 \end{aligned}$$

Each column in the table has the same number of ones and zeros, and hence no bias. However many of the possible Z values have different numbers, and hence do have bias.

For a 4-bit S-Box there are 256 combinations of variables

The $N_L(X, Y)$ values are the number of zeros generated by each particular combination of input and output (in this case $N_L(1, 7) = 10$, and $N_L(b, 4) = 8$). A table of all possible 2^8 values is called the linear approximation table, and represents the bias for each constructed variable. This table (like the differential table before) is used in the linear cryptanalytic attack.

Linear approximation table

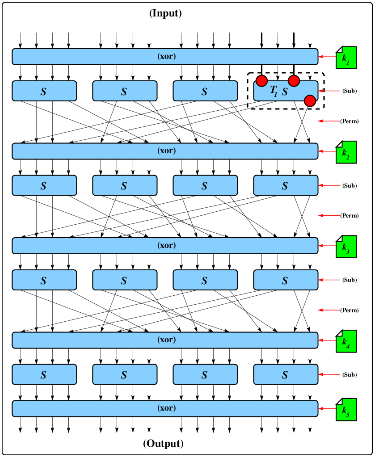
	0	1	2	3	4	5	6	7	8	9	*a*	*b*	*c*	*d*	*e*	*f*
0	16	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
1	8	8	8	4	6	10	10	10	8	12	8	8	10	10	6	10
2	8	6	6	12	10	8	12	10	8	10	6	8	6	8	8	10
3	8	10	10	8	8	6	10	4	8	10	10	8	8	6	10	12
4	8	6	8	10	8	10	8	6	10	8	10	4	10	12	10	8
5	8	10	8	10	10	12	6	8	10	8	10	8	8	6	4	10
6	8	8	6	10	6	10	8	8	6	10	12	12	8	8	10	6
7	8	8	10	10	8	8	10	10	6	6	8	8	14	6	8	8
8	8	8	12	8	8	8	8	12	10	10	10	6	6	6	10	6
9	8	8	8	8	6	10	6	10	10	6	6	10	8	8	12	12
a	8	14	6	8	10	8	8	10	6	8	8	6	8	10	10	8
b	8	10	6	8	8	6	10	8	14	8	8	10	10	8	8	6
c	8	6	8	6	12	6	8	10	8	6	12	10	8	10	8	10
d	8	10	12	10	6	8	10	8	8	6	8	10	6	12	6	8
e	8	8	10	10	10	6	4	8	8	12	6	10	10	10	8	8
f	8	8	10	6	12	12	10	6	8	8	6	10	8	8	10	6

Compute bias using the linear approximation table

For example, to find $\varepsilon(Z_{6,a})$, look at row 6, column a to find $N_L(6,a) = 12$. Subtract 8 from this, and divide by 16 to get $\varepsilon(Z_{6,a}) = +\frac{1}{4}$.

Linear approximation of S-Box

Consider the bias of the dotted-line S-Box. The red-circled bits correspond to X_3, X_1, Y_0, and in this case have a bias of $\varepsilon(Z_{a,1}) = \frac{3}{8}$ (because $N_L(a,1) = 14$).

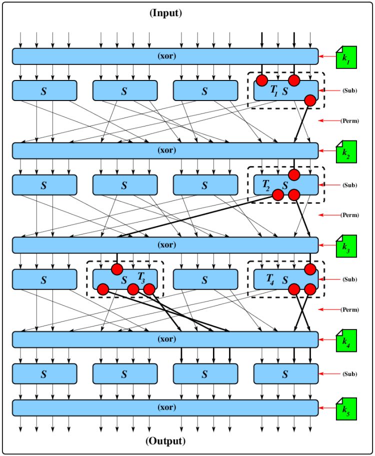


A linear approximation of the S-Box

We do not choose bit sets that have no bias. As an example $\varepsilon(Z_{4,9}) = 0$, and so is not interesting (check for yourself that $N_L(4,9) = 8$).

Linear approximation of S-Box

By carefully choosing S-boxes, and the input and output bits that exhibit bias, we chart a bias from the input of the SPN through to the output. Each set of bits is labelled $T_1, T_2, \ldots$

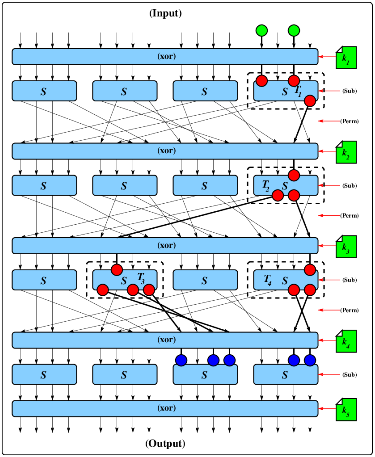


Combining the S-boxes

Consider the calculation of $T_1 \oplus T_2 \oplus T_3 \oplus T_4$. If we xor the dots, note that all the intermediate ones cancel out.

Linear approximation of S-Box

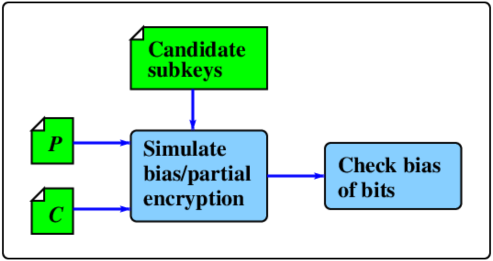
Note that the bias from the red dots/bits $\varepsilon(T_1 \oplus T_2 \oplus T_3 \oplus T_4)$ is exactly the bias of the green and blue bits.



Linear bias of the whole SPN

To use the piling lemma, we assume that T_1, T_2, T_3, T_4 are independent , but of course they are not. However, it is a good approximation.

The main idea...

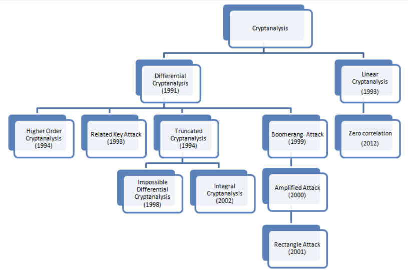


The algorithm starts with the final round subkey

The attacker tries all possible candidate subkeys. For each candidate, using all the pairs of plaintexts-ciphertexts, the attacker looks at the distribution of the output, in particular the bias. The key that generates the same bias is the correct key, since the wrong key likely leads to a bias of 0.

The attack exploits the "probabilistic linear relationship" among a subset of the input bits and output bits. Note that "xor" of bits can be viewed as a "linear" combination of the bits.

Integral cryptanalysis


Cryptanalysis
Differential Cryptanalysis (1991)
Linear Cryptanalysis (1993)
Higher Order Cryptanalysis (1994)
Related Key Attack (1993)
Truncated Cryptanalysis (1994)
Boomerang Attack (1999)
Zero correlation (2012)
Impossible Differential Cryptanalysis (1998)
Integral Cryptanalysis (2002)
Amplified Attack (2000)
Rectangle Attack (2001)

A chosen plaintext attack (sets of plaintexts)

Instead of using pairs of plaintexts as in differential cryptanaysis, this uses the properties of particular sets of encryptions. The Rijndael authors were working on the “Square” cipher, designed to be resistant to differential and linear cryptanalysis with only four rounds, but a colleague devised an attack on Square. Square led to Rijndael which led to AES.

The attack exploits the (largely) bijective nature of these classes of SPNs. Each round modifies the state, represented (in the Rijndael/AES case) as a 4x4 array of bytes. The properties of sets of these bytes are of interest. For example, the XOR of the set of all bytes ($00 - FF$) is the value $Z = 0$.

The big idea in integral cryptanalysis

Properties of sets of words in AES (4×4) state:

$$\begin{array}{|c|c|c|c|}\hline \mathcal{A} & \mathcal{C} & \mathcal{C} & \mathcal{C} \\ \hline \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} \\ \hline \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} \\ \hline \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|c|}\hline \mathcal{A} & \mathcal{C} & \mathcal{C} & \mathcal{C} \\ \hline \mathcal{A} & \mathcal{C} & \mathcal{C} & \mathcal{C} \\ \hline \mathcal{A} & \mathcal{C} & \mathcal{C} & \mathcal{C} \\ \hline \mathcal{A} & \mathcal{C} & \mathcal{C} & \mathcal{C} \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|c|}\hline \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \\ \hline \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \\ \hline \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \\ \hline \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \\ \hline \end{array} \rightarrow$$

$$\begin{array}{|c|c|c|c|}\hline \mathcal{Z} & \mathcal{Z} & \mathcal{Z} & \mathcal{Z} \\ \hline \mathcal{Z} & \mathcal{Z} & \mathcal{Z} & \mathcal{Z} \\ \hline \mathcal{Z} & \mathcal{Z} & \mathcal{Z} & \mathcal{Z} \\ \hline \mathcal{Z} & \mathcal{Z} & \mathcal{Z} & \mathcal{Z} \\ \hline \end{array}$$

We choose the 256 16-byte (128-bit) plaintexts, with all 256 values of the first byte/word, and all the rest of the bytes fixed. On the left we see the initial state, with the properties for each word ($\mathcal{A}$ active, and $\mathcal{C}$ constant).

After one round, the MixColumn step forces the first column to be all active, and after the second round, the ShiftRows and MixColumns step force all words to be in play. After the third round, the sums are zero.

Integral cryptanalysis

With the first four rounds, we have...

Having gathered the plaintext and output of this truncated round encryption, we can trial all 256 keys for the first (8-bit) word of the state - one of them will result in the property $\mathcal{Z}$ for all 16 bytes. We can repeat this experiment with all 16 words, or patterns of such words.

This attack led directly to the requirement to extend the number of rounds for Square to 6 or more, and led to the choices of rounds for Rijndael and hence AES.

Finally...

In a sense, none of these cryptanalytic techniques change the nature of symmetric encryption schemes. If the AES family is a "computationally indistinguishable" cipher, then none of these attacks make it not such a cipher. An attack may now be possible on (say) AES-128, but the attack is still exponential in (an extended) key size. If we consider the parameter to be a linear function of keysize, number of rounds, and word/block size, it is still exponential, and so we can harden these types of ciphers so that they are once again computationally indistinguishable.

Why primes?

In the book "Contact"...

...the heroine recognizes an alien communication because it starts...

2.. 3.. 5.. 7.. 11.. 13.. 17.. 19.. 23...[a]

[a] Actually, in the book, Sagan used 1,2,3,5... :)

Is it just a coincidence that...

... the numbers on the main Real Madrid player's jerseys were: Carlos, No 3; Zidane, No 5; Raul, No 7; Owen, No 11?

Why primes?

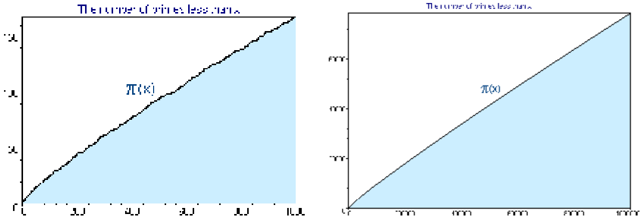
For 2500 years mathematicians studied prime numbers just because they were interesting, without any idea they would have practical applications.

Possible real-world uses:

1. A prime number of ball bearings to reduce wear.
2. Possibly... the 13 and 17-year periodic emergence of cicadas may be due to coevolution with predators (that lost and became extinct).

We do not know when the next one will occur.

But ... we do know that the density is predictable...



(Asymptotic to $\frac{x}{\log x}$)

Why primes?

- Because 2500 years of mathematics has failed to uncover some basic prime properties, they make a good candidate for constructing difficult (impossible to decrypt) translations... and hence our interest in them...
- Because primes are beautiful...

Consider the following problem:

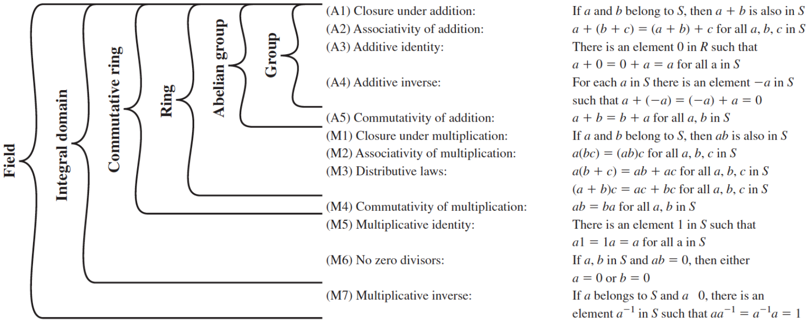
Question: *Is it possible to find an arbitrary sized sequence of numbers that are* $\underline{\textit{not}}$ *primes?*

Answer: *YES!*

How to get *n* not-primes in a row:

- If you want 3 not-primes in a row, calculate $4 * 3 * 2 * 1 = 4!$, and choose the numbers $4! + 2$, $4! + 3$ and $4! + 4$. None can be a prime.
- If you want $42,000$ not-primes in a row, calculate $42001 * \ldots * 2 * 1 = 42001!$, and choose the numbers $42001! + 2$, $42001! + 3$... None can be a prime.
- If you want 4847584765843775375983487509485945495840 not-primes ...

Abstract algebra concepts...



(A1) Closure under addition:	If a and b belong to S, then $a + b$ is also in S
(A2) Associativity of addition:	$a + (b + c) = (a + b) + c$ for all a, b, c in S
(A3) Additive identity:	There is an element 0 in R such that $a + 0 = 0 + a = a$ for all a in S
(A4) Additive inverse:	For each a in S there is an element $-a$ in S such that $a + (-a) = (-a) + a = 0$
(A5) Commutativity of addition:	$a + b = b + a$ for all a, b in S
(M1) Closure under multiplication:	If a and b belong to S, then ab is also in S
(M2) Associativity of multiplication:	$a(bc) = (ab)c$ for all a, b, c in S
(M3) Distributive laws:	$a(b + c) = ab + ac$ for all a, b, c in S $(a + b)c = ac + bc$ for all a, b, c in S
(M4) Commutativity of multiplication:	$ab = ba$ for all a, b in S
(M5) Multiplicative identity:	There is an element 1 in S such that $a1 = 1a = a$ for all a in S
(M6) No zero divisors:	If a, b in S and $ab = 0$, then either $a = 0$ or $b = 0$
(M7) Multiplicative inverse:	If a belongs to S and a 0, there is an element a^{-1} in S such that $aa^{-1} = a^{-1}a = 1$

(Diagram from Stallings, a crypto book)

Fields, rings and groups

In crypto, we often work with finite, cyclic mathematical structures. These have been studied for centuries, and it is appropriate for us to know about the mathematical domain in which we must work.

I say *must*, because without knowing the hard facts/limits of the mathematical structures we must deal with, we have *nothing* (!)

17^{th} century Frenchman Pierre de Fermat

Fermat's (little) theorem for $\mathbb{Z}_p$: (primes)

When p is a prime, if a is any non-zero number less than p, then

$$a^{p-1} \bmod p = 1$$

A table showing powers-of-a for $p = 11$:

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}
2	4	8	5	10	9	7	3	6	1
3	9	5	4	1	3	9	5	4	1
4	5	9	3	1	4	5	9	3	1
5	3	4	9	1	5	3	4	9	1
6	3	7	9	10	5	8	4	2	1
7	5	2	3	10	4	6	9	8	1
8	9	6	4	10	3	2	5	7	1
9	4	3	5	1	9	4	3	5	1
10	1	10	1	10	1	10	1	10	1

Fermat's little theorem

Observations

- For $p = 11$ the value is always 1 when the power gets to 10
- Sometimes the value gets to 1 earlier
- Lengths of runs are always numbers that divide evenly into 10
- A value of a for which the whole row is needed is called a *generator*. 2, 6, 7, and 8 are generators.

Simplifying expressions

Because a to a power mod p always starts repeating after the power reaches $p - 1$, you can do this:

$$a^x \bmod p = a^{x \bmod (p-1)} \bmod p$$

Thus modulo p in the expression requires modulo $p - 1$ in the exponent. For $p = 13$, then

$$a^{29} \bmod 13 = a^{29 \bmod 12} \bmod 13 = a^5 \bmod 13$$

Another example result $= 7^{1215}$ mod 13

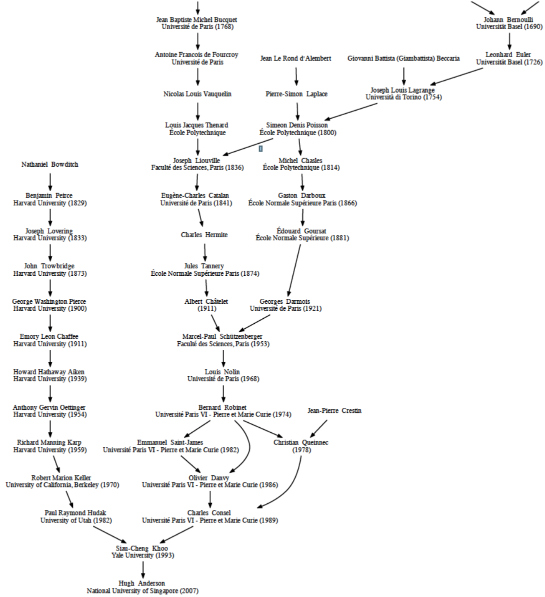
a big number...

6224702750673227370465564559079792689062398648329219130902078771092486991072740587
0651989078101738389949782679348130096777089278266013135577736536148404478380085122
2817392261341421370762400507026834564501614788818580162335818155077291900607338638
1098582099841775377667037286814739670120315712396914000184822340352355906455155667
5341024739645354137741258367626070635933104840329377905370464877106976413186542262
2995052805557584280574185802694213299802280179325494560628948940739344482284649151 1
9714116869895958794732024285742690180232449402567101050831149673563342958092194557
1119113124697462717311124279255445332116504914530077241996189357298508605206780120
8988083552522234194051458556732086842042388893209157040799864871901064991230860288
6575458785483803190210993511026450389154414587258074783062229406697804705969808888
2249767794049127920176330954113185559387768008167786246958079094970578719259627712
7796303487781814106147375370904627195995589087276846994 3 mod 13 = 5

$$
\begin{aligned}
\text{result} &= 7^{1215} \text{ mod } 13 \\
&= 7^{1215} \text{ mod } 13 \\
&= 7^{1215 \text{ mod } 12} \text{ mod } 13 \\
&= 7^{3} \text{ mod } 13 \\
&= 343 \text{ mod } 13 \\
&= 5
\end{aligned}
$$

We can do BIG NUMBER maths without calculating BIG numbers!

Leonhard Euler (1707-1783)



Jean Baptiste Michel Bucquet
Université de Paris (1768)

Johann Bernoulli
Universität Basel (1690)

Antoine Francois de Fourcroy
Université de Paris

Jean Le Rond d'Alembert

Giovanni Battista (Giambattista) Beccaria

Leonhard Euler
Universität Basel (1726)

Nicolas Louis Vauquelin

Pierre-Simon Laplace

Joseph Louis Lagrange
Università di Torino (1754)

Louis Jacques Thenard
École Polytechnique

Simeon Denis Poisson
École Polytechnique (1800)

Nathaniel Bowditch

Joseph Liouville
Faculté des Sciences, Paris (1836)

Michel Chasles
École Polytechnique (1814)

Benjamin Peirce
Harvard University (1829)

Eugène-Charles Catalan
Université de Paris (1841)

Gaston Darboux
École Normale Supérieure Paris (1866)

Joseph Lovering
Harvard University (1833)

Charles Hermite

Édouard Goursat
École Normale Supérieure (1881)

John Trowbridge
Harvard University (1873)

Jules Tannery
École Normale Supérieure Paris (1874)

George Washington Pierce
Harvard University (1900)

Albert Châtelet
(1911)

Georges Darmois
Université de Paris (1921)

Emory Leon Chaffee
Harvard University (1911)

Marcel-Paul Schützenberger
Faculté des Sciences, Paris (1953)

Howard Hathaway Aiken
Harvard University (1939)

Louis Nolin
Université de Paris (1968)

Anthony Gervin Oettinger
Harvard University (1954)

Bernard Robinet
Université Paris VI - Pierre et Marie Curie (1974)

Jean-Pierre Crestin

Richard Manning Karp
Harvard University (1959)

Emmanuel Saint-James
Université Paris VI - Pierre et Marie Curie (1982)

Christian Queinnec
(1978)

Robert Marion Keller
University of California, Berkeley (1970)

Olivier Danvy
Université Paris VI - Pierre et Marie Curie (1986)

Paul Raymond Hudak
University of Utah (1982)

Charles Consel
Université Paris VI - Pierre et Marie Curie (1989)

Siau-Cheng Khoo
Yale University (1993)

Hugh Anderson
National University of Singapore (2007)

Euler's theorem - a generalization of Fermat's

Euler's theorem: (composites)

If N is any positive integer and a is any positive integer less than N with no divisors in common with N, then

$$a^{\phi(N)} \bmod N = 1$$

where $\phi(N)$ is the *Euler phi (totient) function*:

$$\phi(N) = N(1 - 1/p_1)\ldots(1 - 1/p_m)$$

and $p_1, \ldots, p_m$ are all the prime numbers that divide evenly into N.

If N is a prime, then using the formula, we have

$$\phi(N) = N(1 - \frac{1}{N}) = N(\frac{N-1}{N}) = N - 1$$

We see that Fermat's result is a special case of Euler's:

$$a^{\phi(N)} \bmod N = a^{N-1} \bmod N = 1$$

Special cases

Special case #2 (RSA-style composites)

Another special case needed for RSA comes when the modulus is a product of two (different) primes: $N = pq$. Then

$$\phi(N) = N(1 - \frac{1}{p})(1 - \frac{1}{q}) = (p - 1)(q - 1)$$

and so we have

$$a^{(p-1)(q-1)} \bmod pq = 1$$

(if a has no divisors in common with pq and p, q prime)

On the next slide we illustrate Euler with $N = 15$

The table illustrates Euler's theorem for $N = 15 = 3 \times 5$, with

$$\phi(15) = 15(1 - \frac{1}{3})(1 - \frac{1}{5}) = (3 - 1)(5 - 1) = 8$$

Notice here that a 1 is reached when the power is 8, but only for numbers with no divisors in common with 15.

Euler table for $N = 15$, $\phi(N) = 8$

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}
2	4	8	1	2	4	8	1	2	4	8	1	2	4
3	9	12	6	3	9	12	6	3	9	12	6	3	9
4	1	4	1	4	1	4	1	4	1	4	1	4	1
5	10	5	10	5	10	5	10	5	10	5	10	5	10
6	6	6	6	6	6	6	6	6	6	6	6	6	6
7	4	13	1	7	4	13	1	7	4	13	1	7	4
8	4	2	1	8	4	2	1	8	4	2	1	8	4
9	6	9	6	9	6	9	6	9	6	9	6	9	6
10	10	10	10	10	10	10	10	10	10	10	10	10	10
11	1	11	1	11	1	11	1	11	1	11	1	11	1
12	9	3	6	12	9	3	6	12	9	3	6	12	9
13	4	7	1	13	4	7	1	13	4	7	1	13	4
14	1	14	1	14	1	14	1	14	1	14	1	14	1

Arithmetic in the exponent is taken mod $\phi(N)$

If *a* has no divisors in common with *N*,

$$a^x \bmod N = a^{x \bmod \phi(N)} \bmod N.$$

so for example $a^{28} \bmod 15 = a^{28 \bmod 8} \bmod 15 = a^4 \bmod 15$.

Quadratic residues

The square root of $x \in \mathbb{Z}_p$ is... (primes)

...a number $y \in \mathbb{Z}_p$ such that $y^2 \equiv x \mod p$. Note that $\sqrt{2} \mod 7 = 3$, but $\sqrt{3} \mod 7$ doesn't exist.

$\mathbb{Z}_p^*$ is the multiplicative (sub) group and an element $x \in \mathbb{Z}_p^*$ is called a QR (Quadratic Residue) if its square root exists in $\mathbb{Z}_p$. Each $x \in \mathbb{Z}_p^*$ has either 0 or 2 square roots.

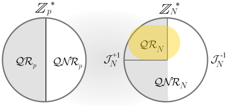
- If a is a square root of $x \mod p$, then so is $-a \mod p$.
- Exactly half of the elements in $\mathbb{Z}_p^*$ are QR.
- a is a QR $(\mathrm{mod}\, p)$ iff $a^{\frac{p-1}{2}} \mod p = 1$ (Euler's criterion).

Computing square roots in $\mathbb{Z}_p^*$ is computationally possible. An $\mathcal{O}(n^4)$ randomized algorithm exists. Easier in some special cases of p, e.g. when $p = 3 \mod 4$, then

$$\sqrt{x} \mod p = x^{\frac{p+1}{4}} \mod 4$$

Given an $x \in \mathbb{Z}_p^*$, deciding whether x is a QR is computationally easy.

Identifying quadratic residues



The Legendre and Jacobi "symbol" notation

Legendre: *identifies* the QR in $\mathbb{Z}_p$; a function of a and p: ($\mathbb{Z}_p$)

$$\left(\frac{a}{p}\right) = a^{\frac{p-1}{2}} \mod p = \begin{cases} 1 & \text{if } a \text{ is a QR} \\ -1 & \text{if } a \text{ is not a QR} \\ 0 & \text{if } a = 0 \mod p \end{cases}$$

Jacobi: is a generalization of Legendre, extending it to $\mathbb{Z}_N$ (composites). ($\mathbb{Z}_N$) Easy to compute given it's prime factorization. For example, given $N = p \times q$ with $p \neq q$, then

$$\left(\frac{a}{N}\right) = \left(\frac{a}{p}\right)\left(\frac{a}{q}\right)$$

Note that there is an algorithm that, given a and N, can find the Jacobi symbol without knowing the factorization of N. From the Jacobi symbol, we can't tell whether a is a QR, but no entry with -1 is a quadratic residue.

Complexity of basic arithmetic

Basic ideas...

We often work with (cyclic) groups in $\mathbb{Z}_p$ and $\mathbb{Z}_N$ with very large p and N.

Computation complexity: the following basic arithmetic operations can be efficiently computed in $\mathbb{Z}_p$ and $\mathbb{Z}_N$:

Operations	Complexity
Addition	$\mathcal{O}(m)$
Multiplication	$\mathcal{O}(m^2)$
Inverse	$\mathcal{O}(m^2)$
Exponentiation	$\mathcal{O}(m^3)$

Programming issues: Since our machines work with 32 or 64-bit integers, we have libraries such as the GMP (GNU Multiple Precision) library, and Java “BigInteger” classes.

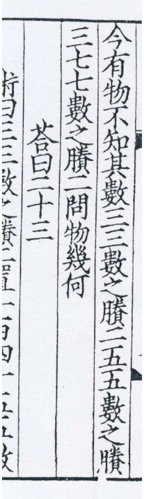
On finding generators:

```
gens=[]
for s in facs:
   g=2
   generators=[]
   while g<s[0]:
      gen=1
      for j in s[1]:
         if (g**((s[0]-1)/j))%s[0]==1:
            gen=0
            break
      if gen:
         generators.append(g)
      g=g+1
   gens.append([s[0],generators])
```

Algorithm to find a generator modulo p:

1. Find prime factors $q_1, q_2, \ldots$ of $\phi(p)$.
2. Select a random candidate a.
3. For each factor q_i, calculate $a^{\frac{p-1}{q_i}} \bmod p$. If any one is 1, then back to step 2.
4. Otherwise a is a generator.

CRT - the Chinese Remainder Theorem



今有物不知其數三三數之賸二五五數之賸三七七數之賸二問物幾何

荅曰二十三

術曰三三數之賸二置一百四十五五數

Problem posed by Wei/Jin mathematician Sun Tzu 1700 years ago:

There are certain things whose number is unknown. Repeatedly divided by 3, the remainder is 2; by 5 the remainder is 3; and by 7 the remainder is 2. What will be the number?

The Chinese remainder theorem - nowadays

Two simultaneous congruences $n = n_1 \bmod m_1$ and $n = n_2 \bmod m_2$ are only solvable when $n_1 = n_2 \bmod (\gcd(m_1, m_2))$. The solution is unique modulo $\operatorname{lcm}(m_1, m_2)$.

It demonstrates to us that a number less than the product of two primes can be uniquely identified by its residue modulo those primes. It is useful in RSA.

CRT - the Chinese Remainder Theorem

Worked example

The original problem:

$$\begin{aligned} x &= 2 \bmod 3 \qquad (1)\\ x &= 3 \bmod 5 \qquad (2)\\ x &= 2 \bmod 7 \qquad (3) \end{aligned}$$

From (1), we know that $x = 3n + 2$ for some n. Substituting this in (2) gives $3n = 1 \bmod 5$. This reduces to a simpler equation $n = 2 \bmod 5$ which is equivalent to $n = 5m + 2$ for some m. Substituting this back into $x = 3n + 2$ gives us $x = 15m + 8$. Substituting this in (3) gives $15m + 8 = 2 \bmod 7$, or $m = 1 \bmod 7$, i.e. $m = 7o + 1$.

From this we can see that $x = 105o + 23$. Note that $105 = \mathrm{lcm}(3, 5, 7)$ and the solutions are $23, 128$ (and so on).

Euclidean algorithm

Some theory to compute the multiplicative inverses.

To find the **gcd** of two numbers (say 2394 and 154), use the prime factorization:

$$\begin{aligned} 2394 &= 2 * 3 * 3 * 7 * 19 \\ 154 &= 2 * 7 * 11 \\ \therefore \text{gcd}(2394, 154) &= 2 * 7 = 14 \end{aligned}$$

Euclidean algorithm

Unfortunately, it is not easy to find the prime factors of integers. The **gcd** of two integers can however be found by repeated application of division, using the Euclidean algorithm. You repeatedly divide the divisor by the remainder until the remainder is 0. The **gcd** is the last non-zero remainder.

Example:

$$\begin{aligned} \text{dividend} &= \text{quotient} * \text{divisor} + \text{remainder} \\ 2394 &= 15 * 154 + 84 \\ 154 &= 1 * 84 + 70 \\ 84 &= 1 * 70 + 14 \\ 70 &= 5 * 14 \\ \therefore \text{gcd}(2394, 154) &= 14 \end{aligned}$$

Extended Euclidean algorithm

An interesting property

If the $\gcd(a, b) = r$ then there exist integers m and n so that $ma + nb = r$. Use to calculate the multiplicative inverse of an element x modulo n.

The extended Euclidean algorithm

1. We begin by dividing n by x, and as we carry out each step i of the Euclidean algorithm discovering the quotient q_i, we also calculate an extra number x_i. For the first two steps $x_0 = 0$ and $x_1 = 1$.
2. For the following steps, calculate $x_i = x_{i-2} - x_{i-1}q_{i-2} \bmod n$.
3. If the last non-zero remainder occurs at step k, then if this remainder is 1, x has an inverse and it is x_{k+2}.

The inverse of 15 modulo 26, showing the method:

0 :	26	$=$	$1 * 15 + 11$	$x_0 = 0$		
1 :	15	$=$	$1 * 11 + 4$	$x_1 = 1$		
2 :	11	$=$	$2 * 4 + 3$	$x_2 = 0 - 1 * 1 \bmod 26$	$=$	25
3 :	4	$=$	$1 * 3 + 1$	$x_3 = 1 - 25 * 1 \bmod 26$	$=$	2
4 :	3	$=$	$3 * 1 + 0$	$x_4 = 25 - 2 * 2 \bmod 26$	$=$	21
				$x_5 = 2 - 21 * 1 \bmod 26$	$=$	7

Easy problems in $\mathbb{Z}_p$

(...assuming p is very large)

Basic math: Generating a random element, adding and multiplying elements, finding inverse, exponentiation.

QR: Testing if an element is a QR and computing its square root mod p if it is QR.

DDH: Decisional Diffie-Hellman problem[a]: Let g be a generator of a cyclic group $\mathcal{G}$. Given a positive integer 3-tuple (a, b, z), distinguish with probability greater than 0.5 whether the three outputs are from process A or B:

A: Output (g^a, g^b, g^z)
B: Output (g^a, g^b, g^{ab})

[a] For the group $\mathbb{Z}_p^*$, DDH is not hard (...use the Jacobi symbol).

Believed to be hard problems in $\mathbb{Z}_p$

But anything can change...

DL: Discrete log problem: Let g be a generator of a cyclic group. Given x, find r such that $x = g^r$.

CDH: Computational Diffie-Hellman problem: Let g be a generator of a cyclic group. Given g^a, g^b, find g^{ab}.

Cryptosystems based on the hardness of these problems:

1. Diffie-Hellman key exchange agreement
2. Elgamal public key encryption
3. DL-based digital signatures like Elgamal, DSA, DSS...

Easy problems in $\mathbb{Z}_N$

(...assuming $N = p \times q$, and p, q are large primes)

Basic math: Generating a random element, adding and multiplying elements, finding inverse, exponentiation.

Jacobi: Finding the Jacobi symbol.

Hard problems in $\mathbb{Z}_N$

These get easier if factorization is known...

QR: Testing if an element is a QR in $\mathbb{Z}_N$.
(Computing the Jacobi symbol cannot solve the problem).

SQRT: Computing the square root of a QR in $\mathbb{Z}_N$.
Provably as hard as factoring N.
Rabin cryptosystems based on this hardness.

$\phi(N)$: Computing $\phi(N)$ - Provably as hard as factoring N. If you can efficiently compute $\phi(N)$, then you can also factor N.

Roots: Computing the e-th roots mod N when $gcd(e, N) = 1$.
RSA cryptosystems based on this hardness.

These do not get easier if factorization is known...

DL: Discrete log problem With g being a generator of $\mathbb{Z}_N^*$, then given $x \in \mathbb{Z}_N^*$, find an r such that $x = g^r \mod N$.

CDH: Computational Diffie-Hellman problem With g being a generator of $\mathbb{Z}_N^*$, then given $g^a, g^b \in \mathbb{Z}_N^*$, find $g^{ab} \mod N$.

Presented as *one-way* functions

Three examples. Assume $N = p \times q$ (two large primes)

DL: Discrete Logarithm: With prime p and an element $g \in \mathbb{Z}_p^*$ of large order, $f(x) = g^x \bmod p$. It is linear: Given $a \in \mathbb{Z}$, $f(x)$ and $f(y)$, it is easy to compute $f(ax)$ and $f(x+y)$. Its one-wayness is essential for the Diffie-Hellman key exchange protocol and Elgamal public key systems.

RSA: Let e be an integer such that $gcd(e, N) = 1$, $f(x) = x^e \bmod N$. Note that given the factorization of N, the function f can be inverted efficiently. The one-wayness property is essential for the RSA public key system.

Rabin: $f(x) = x^2 \bmod N$ This function is one-way if there is no efficient algorithm to factor N. The function can be inverted efficiently if the factorization of N is known. The one-wayness property is essential for Rabin's scheme.

Summary

The framework for asymmetric systems...

- Multiplicative inverses may not exist in modulo arithmetic. For a modulus N and a number x, x^{-1} exists iff $gcd(x, N) = 1$.
- CRT gives a mapping from $\mathbb{Z}_N$ to $\mathbb{Z}_p \times \mathbb{Z}_q$.
- Some problems are easy in $\mathbb{Z}_p$ but the same problems become difficult in $\mathbb{Z}_N$ if the factorization of N is unknown.
- Some problems, in particular Discrete Log, are hard in both.
- Factorization and Discrete Log are the two main types of problems. Many cryptosystems depend on their “hardness”.