

Generating Random Numbers

- ∅ We need to generate a sequence that looks random but is reproducible
- ∅ There shouldn't be any obvious regularities, otherwise Eve can learn the pattern after seeing several numbers, and guess the next ones
- ∅ We would like to cover the whole range of numbers (e.g. 2^n if the number has n bits)

1

Linear Congruential Generators

- ∅ Generators of the form

$$X_n = (aX_{n-1} + b) \bmod m$$

- ∅ A period of a generator is number of steps before it repeats the sequence
- ∅ If a , b and m are properly chosen, this generator will be *maximal period generator* and have period of m
- ∅ It has been proven that any polynomial congruential generator can be broken

2

Linear Feedback Shift Registers

- ∅ Used for cryptography today
- ∅ A **shift register** is transformed in every step through **feedback function**
 - ∅ Contents are shifted one bit to the right, the bit that "falls out" is the output
 - ∅ New leftmost bit is XOR of bits in the shift register, **tap sequence**
 - ∅ If we choose a proper tap sequence period will be $2^n - 1$

3

Linear Feedback Shift Registers

$$X^4 = X^4 \oplus X^1$$

1111		0110	1	1000	1
0111	1	0011	0	1100	0
1011	1	1001	1	1110	0
0101	1	0100	1	1111	0
1010	1	0010	0		
1101	0	0001	0		

4

Linear Feedback Shift Registers

- ∅ Proper tap sequences are those where a polynomial from a tap sequence + 1 is a primitive polynomial mod 2
- ∅ A primitive polynomial of degree n is an irreducible polynomial that divides $X^{2^n-1} + 1$ but does not divide $X^d + 1$ for any d that divides $2^n - 1$

5

Linear Feedback Shift Registers

- ∅ There are tables of primitive polynomials
- ∅ LFSR is fast in hardware but slow in software
- ∅ Internal state for LFSR is the next n output bits
- ∅ Eve can observe those bits and learn the feedback function after $2n$ steps
- ∅ LFSR are building blocks in encryption algorithms

6

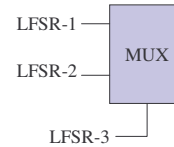
Combining LFSR

- ∅ Use several LFSR with different lengths and primitive polynomials
- ∅ Combine them in non-linear fashion and use the output as random sequence

7

Geffe Generator

- ∅ Use 3 LFSRs, two are input into a multiplexer and third selects the output bit



- ∅ Can be generalized to n LFSRs
- ∅ Easily broken

8

Alternating Stop-and-Go Generator

- ∅ Use 3 LFSRs of different length
 - ∅ LFSR-2 is clocked when the output of LFSR-1 is 1
 - ∅ LFSR-3 is clocked when the output of LFSR-1 is 0
 - ∅ Output is XOR between LFSR-2 and LFSR-3
- ∅ Currently thought secure

9

Gollmann Cascade

- ∅ Use k LFSRs of similar (or same) lengths
 - ∅ LFSR-2 is clocked when the output of LFSR-1 is 1
 - ∅ LFSR-3 is clocked when the output of LFSR-2 is 1
 - ∅ etc.
- ∅ Currently thought secure if number of LFSR is large even if they are short

10

Shrinking Generator

- ∅ Use 2 LFSRs
 - ∅ Clock both
 - ∅ If the output of LFSR-1 is 1 use the output of LFSR-2
 - ∅ Otherwise throw away both bits and clock again, output nothing
- ∅ Currently thought secure

11

Additive Generators

- ∅ Produce random words instead of random bits
- ∅ Register holds sequence of words $X_{m-1}X_{m-2}...X_0$, all n -bit long

$$X_i = (X_{i-a} + X_{i-b} + \dots + X_{i-m}) \bmod 2^n$$

- ∅ If a, b, \dots, m are chosen right the period of the generator is $2^n - 1$
- ∅ For example using a, b, \dots from the table of primitive polynomials should work

12

Pike

- ∅ Use 3 additive generators, for example
 - $A_i = (A_{i-55} + A_{i-24}) \bmod 2^{32}$
 - $B_i = (B_{i-57} + B_{i-7}) \bmod 2^{32}$
 - $C_i = (C_{i-58} + C_{i-19}) \bmod 2^{32}$
- ∅ Look at addition carry bits, if all three are 0 or 1 then clock all three generators, otherwise clock the two that agree; save carry bits for next time
- ∅ Final output is XOR of the outputs of all three generators
- ∅ Currently thought secure

13

Mush

- ∅ Use 2 additive generators, for example
 - $A_i = (A_{i-55} + A_{i-24}) \bmod 2^{32}$
 - $B_i = (B_{i-57} + B_{i-7}) \bmod 2^{32}$
- ∅ If carry bit of A is set, clock B
- ∅ If carry bit of B is set, clock A
- ∅ Clock A and set the carry bit, if there is carry
- ∅ Clock B and set the carry bit, if there is carry
- ∅ Final output is XOR of A and B outputs
- ∅ Currently thought secure

14

Other Cool Crypto Stuff

- ∅ **Broadcasting a secure message**
 - ∅ There are n people in the group, we know we will communicate with subsets but don't know which ones in advance
 - ∅ Strawman's approach generates a lot of keys ($2^n - 2$) or encrypts a message separately to everyone in the subset
 - ∅ Multiple-key public-key cryptography solves this problem

15

Broadcasting a Secure Message

- ∅ Imagine a variant of public key cryptography with 3 keys, one encrypts the message, the other two are needed to decrypt it

Key distribution	
Alice	K_A
Bob	K_B
Carol	K_C
Dave	K_A and K_B
Ellen	K_B and K_C
Frank	K_A and K_C

16

Broadcasting a Secure Message

- ∅ This scheme only needs n keys
- ∅ Problem is that we now have to tell everyone which keys we have used, or who are intended recipients

17

Broadcasting a Secure Message

- ∅ Alice wants to broadcast a message to a subset of listeners; only people intended to receive the message can read it, everyone else receives garbage
- ∅ Alice shares a key with every listener
- ∅ Alice chooses random key K and encrypts message with K , then encrypts K with keys of intended listeners
- ∅ Every listener tries to decrypt K and read the message

18

Broadcasting a Secure Message

- ∅ Each listener shares a key with Alice, all pairwise keys are relatively prime
- ∅ Alice encrypts the message in K, then computes R such that $R \equiv K \pmod{K_i}$ for every intended listener i and $R \equiv 0 \pmod{K_i}$ otherwise
- ∅ She sends the ciphertext and R
- ∅ Listeners attempt to recover K

19

Other Cool Crypto Stuff

∅ Secret Splitting

- ∅ Trent has invented a new beverage formula. He would like to be sure that none of his workers can go to competition and sell it
- ∅ He splits the formula into n pieces and gives each piece to a worker
- ∅ What if pieces carry some partial information?

20

Secret Splitting

- ∅ Trent chooses $n-1$ random keys S_i same length as the message
- ∅ He performs the step:

$$M \oplus S_1 \oplus S_2 \oplus \dots \oplus S_{n-1} = S_n$$
- ∅ He gives S_i to his workers
- ∅ Problem: what if Trent goes on holiday and is shipwrecked and Alice is fired?

21

Secret Splitting

- ∅ Threshold scheme solves the problem when we want to divide the message into n pieces so that $m, m < n$ people can reconstruct it
 - ∅ This is called (m,n) threshold scheme
 - ∅ Pieces are called **shadows**

22

Generating Shadows – LaGrange Interpolating Polynomial Scheme

- ∅ Choose a prime p which is larger than n and also larger than any message you might wish to split
- ∅ Generate an arbitrary polynomial of degree $m-1$

$$\text{poly}(x) = (ax^{m-1} + bx^{m-2} + \dots + mx + M) \pmod{p}$$
- ∅ Evaluate polynomial at n points – these values will be shadows
- ∅ Hand out shadows and p , forget coefficients

23

Advanced Threshold Schemes

- ∅ You want one person to be more important than others?
 - ∅ Give her more shadows
- ∅ You want two hostile delegations to be able to reconstruct the secret only if 2 people from one group and 3 people from another group agree
 - ∅ Create a polynomial which is a product of a linear and a quadratic equation, evaluate these equations and hand out shadows to two groups

24

Other Cool Crypto Stuff

∅ **Bit Commitment**

- ∅ Alice claims she can guess the winner in a horse race. She would like to prove it to Bob without revealing her guess. Bob would like to make sure she didn't change her prediction

25

Bit Commitment

- ∅ Bob generates a random bit string R and sends it to Alice
- ∅ Alice creates a message consisting of R and her prediction and encrypts this with random key K and sends it to Bob
- ∅ If Bob did not use R, Alice could cheat

26

Flipping a Fair Coin

- ∅ Alice and Bob want to flip a coin fairly over the network
 - ∅ Alice flips a coin, Bob guesses the flip
 - ∅ Alice must not be able to re-flip the coin
 - ∅ Bob must not be able to derive the value she flipped before guessing
- ∅ Bit-commitment scheme can help us here

27

Other Cool Crypto Stuff

∅ **One-way accumulators**

- ∅ Alice is a spy and occasionally she has to meet other spies in bars
- ∅ Spies have to be able to verify that they work for the same agency but they don't want to carry a membership list
- ∅ They also don't want to carry ID cards

28

One Way Accumulators

- ∅ Like one-way hash function but commutative
- ∅ Alice calculates the accumulation of every name except hers and carries this around
- ∅ Bob does the same
- ∅ Nonmembers can be given accumulation of everybody
- ∅ Example: $A(x_i, y) = x_{i-1}^y \bmod n$
 - ∅ X_0 and n must be agreed upon in advance

29

Other Cool Crypto Stuff

∅ **Quantum cryptography**

- ∅ Alice and Bob communicate a shared secret
- ∅ If Eve eavesdrops she disturbs the message

30