Information-Theoretic Security: Eve doesn't have enough information to crack the message, even with unlimited computation power

Computation Security: Eve is computationally limited (e.g. can't factor quickly)

Kerkhoff's Principle: assume that

- Eve knows the encryption scheme
- Eve knows the alphabet and language
- Eve doesn't know the key
- Key is chosen randomly

Private Key Encryption

Private Key Encryption requires that $(\forall m \in M)[\text{Dec}(\text{Enc}(m)) = m]$

Shift Cipher

- $\text{Enc}(M, s) = (m_1 + s, m_2 + s, \ldots)$
- $\text{Dec}(C, s) = (c_1 - s, c_2 - s, \ldots)$
- Cracked using **isEnglish** and **frequency analysis**: $f_E \cdot f_E \approx$ 0.065$ all others will be ≈ 0.038

Affine Cipher: $K = \{(a, b) \mid 0 \leq a, b \leq 25 \text{ and } a \text{ rel prime } 26\}$ - Encrypt: $x \to ax+b \pmod{26}$ - Decyrpt: $x \to a^{-1}(x-b) \pmod{26}$ - Only requirement is that we need $\gcd(a, 26) = 1$ so that a^{-1} exists - To crack, test all possible $(a, b) \in K$ and do frequency analysis. Take the largest $f_E \cdot f_{a,b}$

Quadratic Cipher: $K = \{(a, b, c) \mid 0 \leq a, b, c \leq 25\}$

- Encrypt: $x \to ax^2 + bx + c$
- Need to ensure that $f(x)$, with a, b, c has an inverse, but this is hard b/c we need to test each individual $f(0) \ldots f(25)$

General Sub Cipher: idea is to take a permutation of $\{0, \ldots, 25\}$ as the key

- Enc: $x \to f(x)$
- Dec: $x \to f^{-1}(x)$

Random Looking Cipher (Keyword Shift Cipher): create a random looking permutation

- Key is `(phrase, shift)`
- Permutation is generated by
  - writing out phrase (removing duplicate letters)
  - horizontally shifting the text by `shift`
- Main concern is that for a small phrase, there are a long sequence of consecutive letters

Vigenere Cipher: key is a word or phrase: $k = (k_1, k_2, \ldots, k_n)$

- $\text{Enc}(m, k) = m_1 + k_1, \ldots, m_n + k_n, m_{k+1} + k_1 \ldots$
- $\text{Dec}(c, k) = c_1 - k_1, \ldots, c_n - k_n, c_{k+1} - k_1 \ldots$
- To crack:
  - Find the length of the key L. Assume that a word that appears frequently will likely appear in the same position $i \pmod{L}$.
    - For example is "aiq" appears in the slots (57, 58, 59), (87, 88, 89), (102, 103, 104), (162, 163, 164), can deduce that L is a divisor of the gaps between these sequences, $L \in \{1, 3, 5, 15\}$
  - This will create a stream of every Lth character. We can do shift analysis on these streams

One Time Pad: encode messages with a long string of random bits

- $M = \{0, 1\}^n$
- Gen $K = \{0, 1\}^n$
- $\text{Enc}(m, k) = k \oplus m$
- Dec $(c, k) = k \oplus k$

Psuedo Random Bits (Linear Congruential Generator): pick M large and A, B, x_0 random looking and create a recurrence

- $x_{i+1} = Ax_i + B \pmod{M} \qquad \text{need } \gcd(A, M) = 1$

Matrix Cipher: pick an $n \times n$ matrix M, must be invertible

- Enc $x \to M(x)$
- Dec $x \to M^{-1}(y)$
- Easy for Alice and Bob to use since M is small and it's easy to find M^{-1}
- Hard for Eve to brute force since key space is $\approx 26^{n^2}$
- Smart algorithm ($O(n26^n)$): let $T = t_1 t_2 \dots t_N$ where $t_i = t_i^1 \dots t_i^8$. Note that $MT_i = m_i \implies R_j t_i = m_i^j$

Random Shift: Idea is to ensure that the same two messages don't get mapped to the same ciphertext

- Key is a function (e.g. $f(r) = 2r + y$). To encrypt "NYNY"
  - Pick a random $r = 4$ so first shift is $2 * 4 + 7 = 15$
  - Pick a random $r = 10$ so second shift is $2 * 10 + 7 = 1$
  - Pick a random $r = 1$ so third shift is $2 * 1 + 7 = 9$
  - Pick a random $r = 17$ so fourth shift is $2 * 17 + 7 = 15$
  - Send $(4; c)$, $(10; Z)$, $(1; W)$, $(17; N)$

Public Key Encryption

Exponentiation: Input: a, n, p and Output: $a^n \pmod p$

- Convert the exponent into binary
- Use repeated squaring : $x^1, x^2, x^4, \dots$
- Then $x^n = x^{n_1} * x^{n_2} * \cdots$. Where $n_1, n_2, \dots$ are powers of 2

Generator for Z_p^*: for a prime p and $\{g^1, \dots, g^{p-1}\} = \{1, \dots, p-1\}$, g is a **generator** for Z_p^*

- **Theorem**: if g is NOT a generator, then there exists x such that
  - $x \mid p - 1$
  - $x \neq p - 1$
  - $g^x \equiv 1$

```
Input p
Let F be the set of factors of p-1, except p - 1
For g in p/3 to 2p/3:
  Compute g^x for all x in F. If any give 1, then g is NOT a generator
  Otherwise if none of them give 1 for g, then output g
```

- Factoring hard, so extend algorithm for only **safe primes** where $p - 1 = 2q$ for a prime q so $F = \{2, q\}$ and easy to check

Discrete Log Problem: $DL_{p,g}(y) = x$ such that $g^x \equiv y \pmod p$

- Input: $g, a, p \quad 1 \leq g, a \leq p - 1 \quad \langle g \rangle = Z_p^*$
- Output x such that $g^x \equiv a \pmod p$
- A good algorithm would solve this problem in $O(\log(n))$
- In general, we have $g \in \{p/3 \dots, 2p/3\}$ because of some tricks
  - If g is a generator of Z_p^* then $g^{(p-1)/2} \equiv p - 1 \equiv -1$
    * **Example**: $3^x \equiv 92 \pmod{101} \implies 92 \equiv 101 - 9 \equiv (-1)3^2 \equiv 3^{50} * 3^2 \equiv 3^{52}$

Primality Testing

- **Fermat's Little Theorem**: given a prime p, $a^p \equiv a \pmod p$

```
Input p
Choose a random subset R of {2, ..., p-1} of size lg(p)
For each a in R, compute a^p and if not equivalent to a, then p is NOT prime
```

Generating Same Primes of length L:

```
Input L
Pick y in {0, 1}^{L-1}
Let x = 1y
Test if x is a prime and (x-1)/2 is a prime
```

```
If both are prime then output x, else goto step 2
```

- Can be extended to remove multiples of 2 and 3.
- 2 doesn't divide $n \iff (\exists k)[n = 2k+1]$
- 3 doesn't divide $n \iff (\exists k, \exists i \in \{1,2\})[n = 3k+i]$
- Thus $2, 3$ don't divide $n \iff (\exists k, \exists i \in \{1,5\})[n = 6k+i]$

Diffie Hellman Given a security param L

1. Alice finds (p, g) such that $\text{len}(p) = L$
2. Alice sends (p, g) to Bob (Eve sees this)
3. Alice picks random a and sends $g^a \pmod p$ to Bob (Eve sees this)
4. Bob picks random b and sends $g^b \pmod p$ to Alice (Eve see this)
5. Alice computes $(g^b)^a = g^{ab}$
6. Bob computes $(g^a)^b = g^{ab}$

- g^{ab} is the **shared secret** and it believed that it is hard for Eve to find g^{ab}

RSA:

1. Alice picks 2 primes p, q of length L and computes $N = pq$
2. Alice computes $R = \phi(N) = \phi(pq) = (p-1)(q-1)$
3. Alice picks $e \in \{R/3, \ldots, 2R/3\}$ that is relatively prime to R
4. Alice finds d such that $ed \equiv 1 \pmod R$
5. Alice broadcasts (N, e) so that both Bob and Eve can see it
6. Bob wants to send $m \in \{1, \ldots, N-1\}$ and broadcasts $m^e \pmod N$
7. Alice receives $m^e \pmod N$ and computes

- $(m^e)^d \equiv m^{ed} \equiv m^{ed \pmod R} \equiv m \pmod N$

Pollard's ρ Algorithm factor N knowing that p is a factor and $p \leq N^{1/2}$. Idea is to find x, y such that $x \equiv y \pmod p$

- $\gcd(x-y, N)$ will yield a nontrivial factor of N since p divides both
- Idea is to pick random $x_1 \in \{1, \ldots, N-1\}$ and generate $x_i = x_{i-1} * x_{i-1} + c \pmod N$
- Idea is that $x_i = x_j$, then $x_{i+1} = x_{j+a}$, so we find k such that $x_k \equiv x_{2k}$

```
define f(x) = x * x + c

x  = rand(1, N-1),  c = rand(1, N-1),   y = f(x)
while TRUE:
  x = f(x)
  y = f(f(y))
  d = gcd(x - y, N)
  if d != 1 and d != N
    break
output(d)
```

Low e Attack on RSA

- **Chinese Remainder Theorem**:
  - If $N_1, \ldots, N_l$ are relatively prime then there exists $0 \leq x < N_1 \cdots N_L$ such that
  - $x \equiv x_1 \pmod{N_1}$
  - $x \equiv x_2 \pmod{N_2}$
  - $\cdots$
  - $x \equiv x_2 \pmod{N_2}$
- **e Theorem**
  - Instead of $x_1, x_2, \ldots, x_n$, we can look at $m^e \pmod{N_i}$. Then find an x such that $0 \leq x < N_1 \cdots N_L$
  - Finally we have that x is the eth power of m

Same N Attack on RSA:

- **Definition**: A set of numbers if **relatively prime** if no number divides all of them
- **Theorem**: If a, b, c are rel prime, then there exists x_1, x_2, x_3 such that $ax_1 + bx_2 + cx_3 = 1$
- Analysis of generalization of L. Zelda sends m to $A_1, \ldots, A_L$ and Eve sees m^{e_i}
  - $e_1, \ldots, e_L$ are rel prime so there exists $x_1, \ldots, x_L$ such that $\sum_{i=1}^{L} e_i x_i = 1$
  - Eve finds such $x_1, \ldots, x_l$ and computes $(m^{e_1})^{x_1} \times \cdots \times (m^{e_L})^{x_L} = m^{\sum_{i=1}^{L} e_i x_i} \equiv m^1 \equiv m \pmod{N}$

Post Public Key

Learn with Errors Private Key:

- Let $e \in^r A$ mean that e is picked uniformly randomly from the set A
- Let $\frac{p}{4}$ denote $\lfloor \frac{p}{4} \rfloor$ for p odd
- Let $\vec{k}$ denote the key and $\vec{r}$ denote a random vector
- Let γ be a parameter such that we choose e from $\{-\gamma, \ldots, \gamma\}$

Private key $\vec{k}$ Public info p, γ

1. Alice picks a random vector $\vec{r}$
2. Alice computes $\vec{r} \cdot \vec{k} \equiv C \pmod{p}$ and chooses $e \in^r \{-\gamma, \ldots, \gamma\}$
3. Let $D \equiv C + e + \frac{bp}{4}$
4. To send b, Alice sends $(\vec{r}; D)$
5. Bob computes $\vec{r} \cdot \vec{k} \equiv C$
   - $D \approx C \implies b = 0$ (within γ)
   - $D \approx C + \frac{p}{4} \implies b = 1$ (within γ)
   - Otherwise Eve tampered the message

Learn with Errors Public Key:

Idea is for only Alice to have the key vector $\vec{k}$ and have Alice publish noisy equations that satifsfy $\vec{k}$. For $e_i \in^r \{-\gamma, \gamma\}$

- $\vec{r} \cdot \vec{k} \sim C_1 + e_1$
- $\vec{s} \cdot \vec{k} \sim C_2 + e_2$
- $\ldots$

Taking the sum $(r_1 + s_1)x_1 + \cdots + (r_n + s_n) \sim C_1 + C_2 + e_1 + e_2$ so error in $\{-2\gamma, \ldots, 2\gamma\}$

Public information: p, γ, n, m

Alice wants Bob to be able to send $b \in \{0, 1\}$

1. Alice picks a random $\vec{k}$ of length n
2. Alice picks m random $\vec{r}$, each with their own $e \in^r \{-\gamma, \ldots, \gamma\}$

- Let $D = \vec{r} \cdot \vec{k} + e$

3. Alice broadcasts each $(\vec{r}; D)$

- **Note**: $\vec{k}$ satisfies noisy equations and any sum of them

4. Bob wants to send bit b and picks a random uniform set of noisy equations, adds them, and adds $bp/2$ to the solution. Let D' be the sum of all Ds in the selected equations

   $s_1 x_1 + \cdots + s_n x_n \sim D' + bp/2$ if and only if $b = 0$

5. Bob broadcasts $(\vec{s}; F = D' + bp/2)$
6. Alice computes $\vec{s} \cdot \vec{k} - F$

- If small then $b = 0$
- If large then $b = 1$

Psuedorandom Generator (PRG): expands short seed into longer string that looks random

PRG Game: Let p be a polynomial and $G : \{0,1\}^n \to \{0,1\}^{p(n)}$ be computable in poly time

1. Alice picks $x \in \{0,1\}^n$ uniformly and computes $y = G(x) \in \{0,1\}^{p(n)}$
2. Alice picks $z \in \{0,1\}^{p(n)}$ uniformly
3. Alice gives $\{w_1, w_2\} = \{y, z\}$ to Eve (z is either w_1 or w_2)
4. Eve outputs one of $\{w_1, w_2\}$ hoping it's z
5. If Eve outputs z, she wins

Can Eve win this game with probability $\geq 1/2$? Depends on how much Computational Power Eve has

Eve Strategy under Unlimited Computational Power

1. Eve gets w_1, w_z as input (one of which is z)
2. Eve creates the set $A = \{G(x) \mid x \in \{0,1\}^n\}$. This takes exponential time
3. If $w_1 \notin A$, then Eve outputs w_1 and wins
4. If $w_2 \notin A$, then Eve outputs w_2 and wins
5. If $w_1, w_2 \in A$, then Eve outputs w_1, though she might be wrong
   - Probability of Eve losing is $\leq$ probability that $z \in A$.
   - There are $2^{p(n)}$ that z could be, of which 2^n are in A.
   - Thus probability that Eve loses is $\leq \dfrac{2^n}{2^{p(n)}} < 1/2$

However, we restrict Eve to having only polynomial computing time

Threshold Secret Sharing: Zelda has a secret $s \in \{0,1\}^n$

(t, m)-secret sharing is a way for Zelda to give strings to $A_1, \ldots, A_m$ such that

- If any t get together, they can learn s
- If any $< t$ get together, they cannot learn anything

Random String Approach: ($(4,4)$ case) Zelda generates random $r_1, r_2, r_3 \in \{0,1\}^n$ and

- Gives A_1 $s_1 = r_1$
- Gives A_2 $s_2 = r_2$
- Gives A_3 $s_3 = r_3$
- Gives A_4 $s_4 = s \oplus r_1 \oplus r_2 \oplus r_3$

A_1, A_2, A_3, A_4 can recover the secret by doing $s_1 \oplus s_2 \oplus s_3 \oplus s_4 = s$

Polynomial Approach: We can imagine a secret will always be an element of Z_p for prime p

- $s = 20 \implies Z_{23}$
- $s = 23 \implies Z_{23} \implies s = 0$

Zelda wants to send a string to $A_1, \ldots, A_m$ such that

- Any t of $A_1, \ldots, A_m$ can find s
- Any < 1 learn nothing

1. $s \in Z_p$ and Zelda works under mod p
2. Zelda generates random numbers $a_{t-1}, \ldots, a_1 \in Z_p$
3. Zelda creates the polynomial $f(x) = a_{t-1}x^{t-1} + \cdots + a_1 x + s$
4. For $1 \leq i \leq m$, Zelda gives each A_i $f(i)$ (all mod p)
   - Any t people have t points from $f(x)$ and can solve for s
   - Any $< t$ people don't have enough information to figure out s