Public Key Systems

Public Key Systems

- We briefly discuss the following
  - Merkle-Hellman knapsack
  - Diffie-Hellman key exchange
  - Arithmetica key exchange
  - RSA
  - Rabin cipher
  - NTRU cipher
  - ElGamal signature scheme

Public Key Crypto

- Some public key systems provide it all, encryption, digital signatures, etc.
  - For example, RSA
- Some are only for key exchange
  - For example, Diffie-Hellman
- Some are only for signatures
  - For example, ElGamal
- All of these are public key systems

Public Key Systems

- Here we present different systems and mention basic attacks/issues
- In next sections we consider more substantial attacks, namely,
  - Factoring (RSA, Rabin)
  - Discrete log (Diffie-Hellman, ElGamal)
  - RSA implementation attacks

Merkle-Hellman Knapsack

Merkle-Hellman Knapsack

- One of first public key systems
- Based on NP-complete problem
- Original algorithm is weak
  - Lattice reduction attack
- Newer knapsacks are more secure
  - But nobody uses them…
  - Once bitten, twice shy

Knapsack Problem

- Given a set of n weights $W_0, W_1, \ldots, W_{n-1}$ and a sum S, is it possible to find $a_i \in \{0,1\}$ so that

  $$S = a_0 W_0 + a_1 W_1 + \ldots + a_{n-1} W_{n-1}$$

  (technically, this is "subset sum" problem)
- **Example**
  - Weights (62,93,26,52,166,48,91,141)
  - Problem: Find subset that sums to S = 302
  - Answer: 62+26+166+48 = 302
- The (general) knapsack is NP-complete

Knapsack Problem

- General knapsack (GK) is hard to solve
- But **superincreasing knapsack** (SIK) is easy
- In SIK each weight greater than the sum of all previous weights
- **Example**
  - Weights (2,3,7,14,30,57,120,251)
  - Problem: Find subset that sums to S = 186
  - Work from largest to smallest weight
  - Answer: 120+57+7+2 = 186

Knapsack Cryptosystem

1. Generate superincreasing knapsack (SIK)
2. Convert SIK into "general" knapsack (GK)
3. **Public Key:** GK
4. **Private Key:** SIK plus conversion factors

- Easy to encrypt with GK
- With private key, easy to decrypt (convert ciphertext to SIK)
- Without private key, must solve GK ?

Knapsack Cryptosystem

- Let (2,3,7,14,30,57,120,251) be the SIK
- Choose m = 41 and n = 491 with m and n relatively prime, n > sum of SIK elements
- General knapsack

  $2 \cdot 41 \pmod{491} = 82$
  $3 \cdot 41 \pmod{491} = 123$
  $7 \cdot 41 \pmod{491} = 287$
  $14 \cdot 41 \pmod{491} = 83$
  $30 \cdot 41 \pmod{491} = 248$
  $57 \cdot 41 \pmod{491} = 373$
  $120 \cdot 41 \pmod{491} = 10$
  $251 \cdot 41 \pmod{491} = 471$

- General knapsack: (82,123,287,83,248,373,10,471)

Knapsack Example

- **Private key:** (2,3,7,14,30,57,120,251)

  $m^{-1} \bmod n = 41^{-1} \pmod{491} = 12$
- **Public key:** (82,123,287,83,248,373,10,471), n=491
- Example: Encrypt 10010110

  82 + 83 + 373 + 10 = 548
- To decrypt,
  - $548 \cdot 12 = 193 \pmod{491}$
  - Solve (easy) SIK with S = 193
  - Obtain plaintext 10010110

Knapsack Weakness

- **Trapdoor:** Convert SIK into "general" knapsack using modular arithmetic
- **One-way:** General knapsack easy to encrypt, hard to solve; SIK easy to solve
- This knapsack cryptosystem is **insecure**
  - Broken in 1983 with Apple II computer
  - The attack uses **lattice reduction**
- "General knapsack" is not general enough!
- This special knapsack is easy to solve

Lattice Reduction

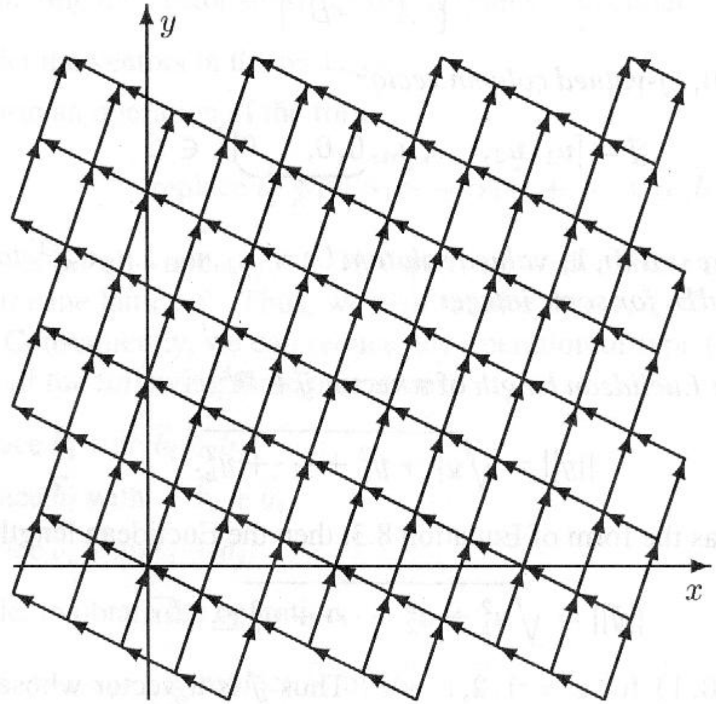
- Many problems can be solved by finding a “short” vector in a **lattice**
- Let $b_1, b_2, \ldots, b_n$ be vectors in $\Re^m$
- All $\alpha_1 b_1 + \alpha_2 b_2 + \ldots + \alpha_n b_n$, each α_i is an integer is a discrete set of points

What is a Lattice?

- Suppose $b_1=[1,3]^T$ and $b_2=[-2,1]^T$
- Then any point in the plane can be written as $\alpha_1 b_1+\alpha_2 b_2$ for some $\alpha_1,\alpha_2 \in \Re$
  - Since b_1 and b_2 are **linearly independent**
- We say the plane $\Re^2$ is **spanned** by (b_1,b_2)
- If α_1,α_2 are restricted to **integers**, the resulting span is a **lattice**
- Then a lattice is a discrete set of points

Lattice Example

- Suppose $b_1=[1,3]^T$ and $b_2=[-2,1]^T$
- The lattice spanned by (b_1,b_2) is pictured to the right



Exact Cover

- **Exact cover** — given a set S and a collection of subsets of S, find a collection of these subsets with each element of S is in exactly one subset
- Exact Cover is a combinatorial problems that can be solved by finding a "short" vector in lattice

Exact Cover Example

- Set S = {0,1,2,3,4,5,6}
- Spse m = 7 elements and n = 13 subsets

Subset:	0	1	2	3	4	5	6	7	8	9	10	11	12
Elements:	013	015	024	025	036	124	126	135	146	1	256	345	346

- Find a collection of these subsets with each element of S in exactly one subset
- Could try all 2^{13} possibilities
- If problem is too big, try **heuristic search**
- Many different heuristic search techniques

Exact Cover Solution

- Exact cover in matrix form
  - Set S = {0,1,2,3,4,5,6}
  - Spse m = 7 elements and n = 13 subsets

Subset:	0	1	2	3	4	5	6	7	8	9	10	11	12
Elements:	013	015	024	025	036	124	126	135	146	1	256	345	346

subsets

elements

$$\begin{bmatrix} 1&1&1&1&1&0&0&0&0&0&0&0&0 \\ 1&1&0&0&0&1&1&1&1&1&0&0&0 \\ 0&0&1&1&0&1&1&0&0&0&1&0&0 \\ 1&0&0&0&1&0&0&1&0&0&0&1&1 \\ 0&0&1&1&0&1&1&0&0&0&1&0&0 \\ 0&1&0&1&0&0&0&1&0&0&1&1&0 \\ 0&0&0&0&1&0&1&0&1&0&1&0&1 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \\ u_{10} \\ u_{11} \\ u_{12} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

m x n   n x 1   m x 1

Solve: AU = B
where $u_i \in \{0,1\}$

Solution:
$U = [0001000001001]^T$

Example

- We can restate AU = B as MV = W where

$$\begin{bmatrix} I_{n\times n} & 0_{n\times 1} \\ A_{m\times n} & -B_{m\times 1} \end{bmatrix} \begin{bmatrix} U_{n\times 1} \\ 1_{1\times 1} \end{bmatrix} = \begin{bmatrix} U_{n\times 1} \\ 0_{m\times 1} \end{bmatrix} \iff AU = B$$

Matrix M Vector V Vector W

- The desired solution is U
  - Columns of M are **linearly independent**
- Let $c_0, c_1, c_2, \ldots, c_n$ be the columns of M
- Let $v_0, v_1, v_2, \ldots, v_n$ be the elements of V
- Then $W = v_0c_0 + v_1c_1 + \ldots + v_nc_n$

Example

- Let L be the lattice spanned by $c_0, c_1, c_2, \ldots, c_n$ (c_i are the columns of M)
- Recall MV = W
  - Where $W = [U,0]^T$ and we want to find U
  - But if we find W, we have also solved it!
- Note W is in lattice L since all v_i are integers and $W = v_0c_0 + v_1c_1 + \ldots + v_nc_n$

Facts

- $W = [u_0, u_1, \ldots, u_{n-1}, 0, 0, \ldots, 0] \in L$, each $u_i \in \{0,1\}$
- The length of a vector $Y \in \Re^N$ is

  $$||Y|| = \mathrm{sqrt}(y_0^2 + y_1^2 + \ldots + y_{N-1}^2)$$

- Then the length of W is

  $$||W|| = \mathrm{sqrt}(u_0^2 + u_1^2 + \ldots + u_{n-1}^2) \leq \mathrm{sqrt}(n)$$

- So W is a very **short** vector in L where
  - First n entries of W all 0 or 1
  - Last m elements of W are all 0
- Can we use these facts to find U?

Lattice Reduction

- If we can find a short vector in L, with first n entries all 0 or 1 and last m entries all 0, then we *might* have found U
  - Easy to test putative solution
- **LLL** lattice reduction algorithm will efficiently find short vectors in a lattice
- Less than 30 lines of pseudo-code for LLL!
- No guarantee LLL will find a specific vector
- But probability of success is often good

Knapsack Example

- What does lattice reduction have to do with the knapsack cryptosystem?
- Suppose we have
  - Superincreasing knapsack

    S = [2,3,7,14,30,57,120,251]
  - Suppose $m = 41$, $n = 491 \Rightarrow m^{-1} = 12 \pmod{n}$
  - Public knapsack: $t_i = 41 \cdot s_i \pmod{491}$

    T = [82,123,287,83,248,373,10,471]
- **Public key:** T **Private key:** (S, m^{-1}, n)

Knapsack Example

- **Public key:** T   **Private key:** (S, m^{-1}, n)
  $S = [2,3,7,14,30,57,120,251]$
  $T = [82,123,287,83,248,373,10,471]$
  $n = 491,\ \ m^{-1} = 12$
- Example: 10010110 is encrypted as
  $82+83+373+10 = 548$
- Then receiver computes
  $548 \cdot 12 = 193 \pmod{491}$
  and uses S to solve for 10010110

Knapsack LLL Attack

- Attacker knows public key

  T = [82,123,287,83,248,373,10,471]

- Attacker knows ciphertext: 548
- Attacker wants to find $u_i \in \{0,1\}$ s.t.

$$82u_0+123u_1+287u_2+83u_3+248u_4+373u_5+10u_6+471u_7 = 548$$

- This can be written as a matrix equation (dot product): $T \cdot U = 548$

Knapsack LLL Attack

- Attacker knows: T = [82,123,287,83,248,373,10,471]
- Wants to solve: T · U = 548 where each $u_i \in \{0,1\}$
  - Same form as AU = B on previous slides
  - We can rewrite problem as MV = W where

$$M = \begin{bmatrix} I_{8\times 8} & 0_{8\times 1} \\ T_{1\times 8} & -C_{1\times 1} \end{bmatrix} = \left[\begin{array}{cccccccc|c} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \hline 82 & 123 & 287 & 83 & 248 & 373 & 10 & 471 & -548 \end{array}\right]$$

- LLL gives us short vectors in the lattice spanned by the columns of M

LLL Result

- LLL finds short vectors in lattice of M
- Matrix M′ is result of applying LLL to M

$$M' = \left[\begin{array}{rrrrrrrr|r}
 & & & \ast & & & & & \\
-1 & -1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\
0 & -1 & 1 & 0 & 1 & -1 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & 0 & 0 & -1 & 1 & 2 \\
1 & -1 & -1 & 1 & 0 & -1 & 0 & -1 & 0 \\
0 & 0 & 1 & 0 & -2 & -1 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & -1 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 \\
\hline
1 & -1 & 1 & 0 & 0 & 1 & -1 & 2 & 0
\end{array}\right]$$

- Column marked with "∗" has the right form
- Possible solution: $U = [1,0,0,1,0,1,1,0]^T$
- Easy to verify this is the plaintext!