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

  $S = a_0W_0 + a_1W_1 + \ldots + a_{n-1}W_{n-1}$

  (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

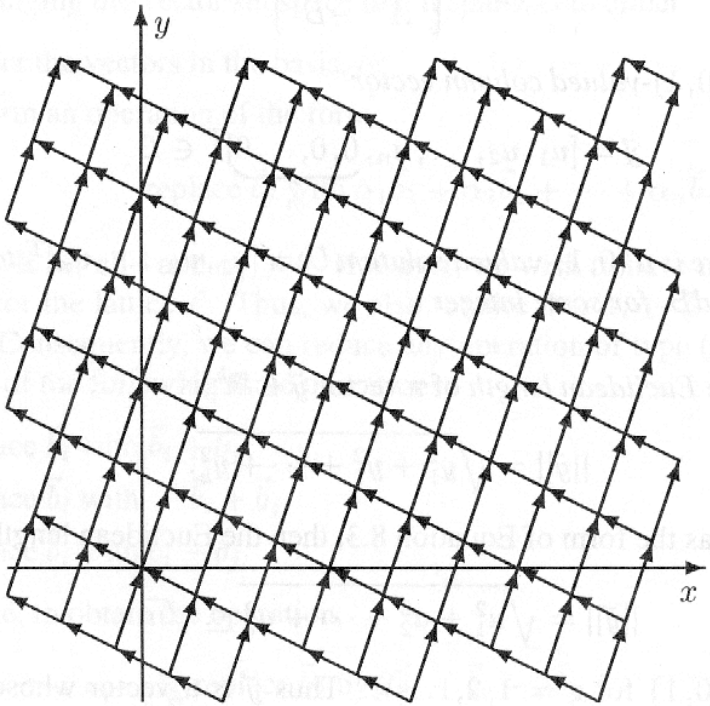
- Many problems can be solved by finding a "short" vector in a **lattice**
- Let $b_1, b_2, \ldots, b_n$ be vectors in $\Re^m$
- All $\alpha_1 b_1 + \alpha_2 b_2 + \ldots + \alpha_n b_n$, each α_i is an integer is a discrete set of points

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

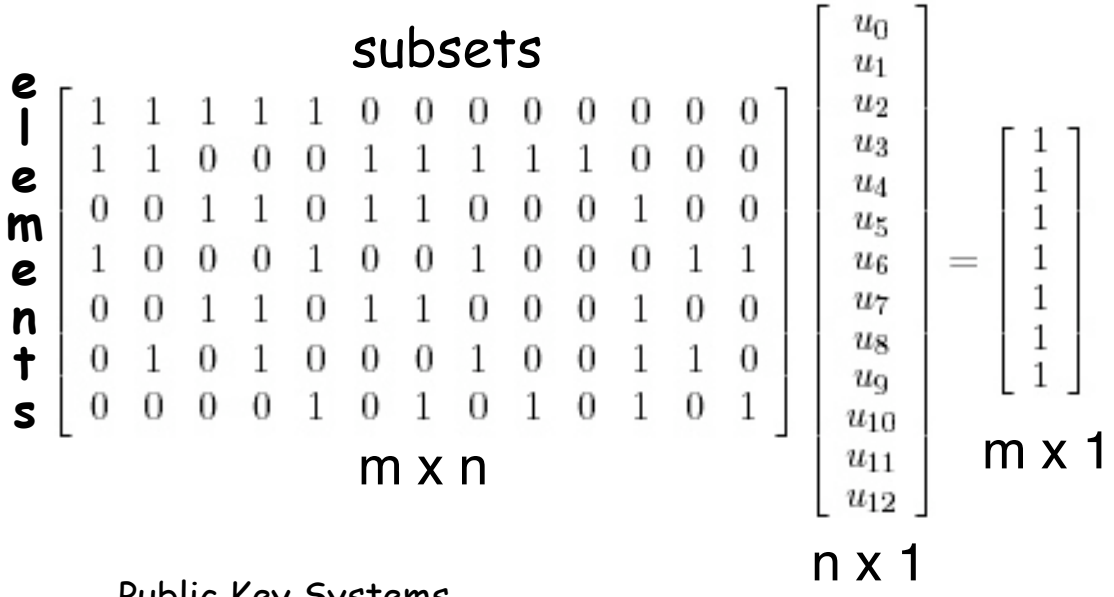
Subset:	0	1	2	3	4	5	6	7	8	9	10	11	12
Elements:	013	015	024	025	036	124	126	135	146	1	256	345	346

- 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



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've 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

  $$\|Y\| = \text{sqrt}(y_0^2 + y_1^2 + \ldots + y_{N-1}^2)$$
- Then the length of W is

  $$\|W\| = \text{sqrt}(u_0^2 + u_1^2 + \ldots + u_{n-1}^2) \leq \text{sqrt}(n)$$
- 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: $T \cdot U = 548$ 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

$$M' = \left[\begin{array}{cccccccc|c} & & & * & & & & & \\ -1 & -1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & -1 & 1 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & -1 & 1 & 2 \\ 1 & -1 & -1 & 1 & 0 & -1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -2 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & -1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 \\ \hline 1 & -1 & 1 & 0 & 0 & 1 & -1 & 2 & 0 \end{array}\right]$$

- 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!

Bottom Line

- Lattice reduction is a surprising method of attack on knapsack
- A cryptosystem is only secure as long as nobody has found an attack
- Lesson: **Advances in mathematics can break cryptosystems**

Diffie-Hellman Key Exchange

Diffie-Hellman Key Exchange

- Invented by Williamson (GCHQ) and, independently, by D and H (Stanford)
- A "key exchange" algorithm
  - To establish a shared symmetric key
- Not for encrypting or signing
- Security rests on difficulty of **discrete log** problem: given g, p, and $g^k \pmod p$, find k

Diffie-Hellman

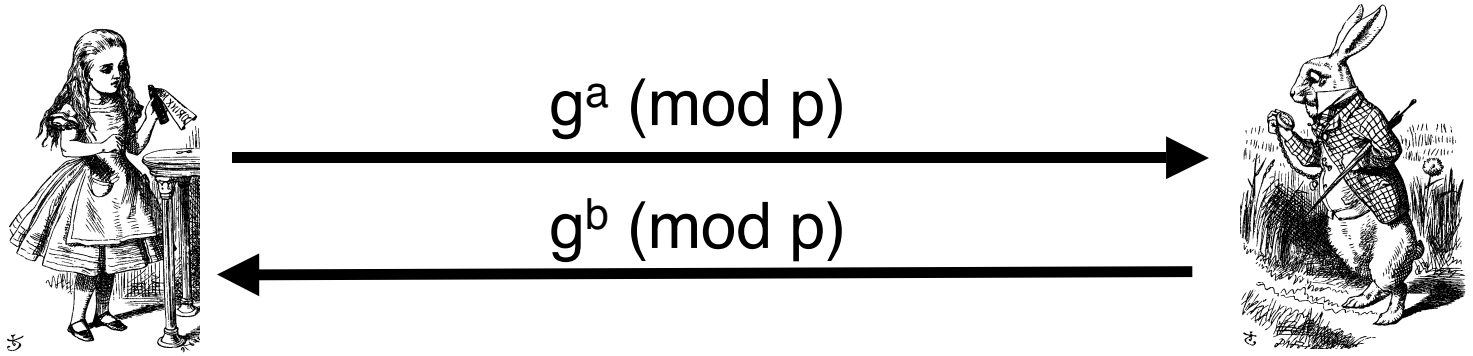
- Let p be prime, let g be a **generator**
  - For any $x \in \{1,2,\ldots,p-1\}$ there is n s.t. $x = g^n \pmod p$
- Alice selects secret value a
- Bob selects secret value b
- Alice sends $g^a \pmod p$ to Bob
- Bob sends $g^b \pmod p$ to Alice
- Both compute shared secret $g^{ab} \pmod p$
- Shared secret can be used as symmetric key

Diffie-Hellman

- Suppose that Bob and Alice use $g^{ab} \pmod p$ as a symmetric key
- Trudy can see $g^a \pmod p$ and $g^b \pmod p$
- Note $g^a g^b = g^{a+b} \neq g^{ab} \pmod p$
- If Trudy can find a or b, system is broken
- If Trudy can solve **discrete log** problem, then she can find a or b

Diffie-Hellman

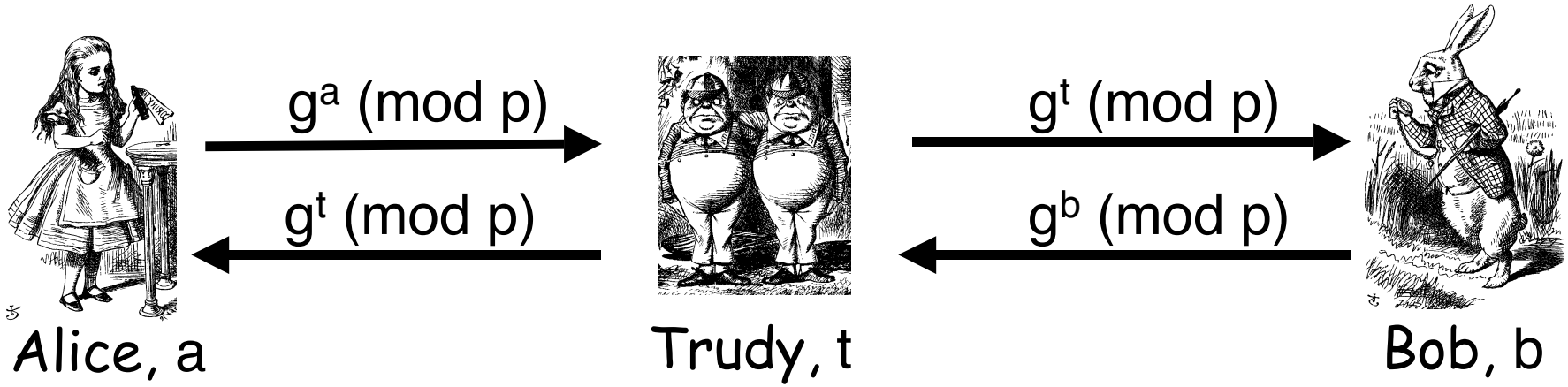
- **Public:** g and p
- **Secret:** Alice's exponent a, Bob's exponent b



- Alice computes $(g^b)^a = g^{ba} = g^{ab}$ (mod p)
- Bob computes $(g^a)^b = g^{ab}$ (mod p)
- Could use $K = g^{ab}$ (mod p) as symmetric key

Diffie-Hellman

- Subject to man-in-the-middle (MiM) attack



- Trudy shares secret $g^{at} \pmod p$ with Alice
- Trudy shares secret $g^{bt} \pmod p$ with Bob
- Alice and Bob don't know Trudy exists!

Diffie-Hellman

- How to prevent MiM attack?
  - Encrypt DH exchange with symmetric key
  - Encrypt DH exchange with public key
  - Sign DH values with private key
  - Other?
- You **MUST** be aware of MiM attack on Diffie-Hellman

Diffie-Hellman Conclusions

- Simple and elegant
- Widely used
- Has several clever uses
  - For example, to make weak PIN-based authentication protocol much stronger
- Man-in-the-middle is serious issue

Arithmetica Key Exchange

Arithmetica Key Exchange

- Relatively new, invented in 1999
- Uses fancy math: group theory
- First, some group theory background
- Then Arithmetica key exchange
- Then simple example
- We mention one attack

Arithmetica Key Exchange

- For example, let G be the set of all finite words from the alphabet $\{1_G, a, b, a^{-1}, b^{-1}\}$
  - Where 1_G is empty word
  - Note that $ab \neq ba$, that is, G is not commutative
  - Not commutative == non-abelian
- Element of G include

  $abaab^{-1}b^{-1}$, $bba^{-1}a1_Gba$, $bbbb$
- Apply properties of exponents to simplify:

  aba^2b^{-2}, b^3a, b^4

Arithmetica Key Exchange

- Define binary operation "$*$" on G
- The operation is concatenation
- For example, $aba^2b^{-2} * b^3a = aba^2ba$
- The set G with "$*$" is a **group**
  - The free group on two generators
- We write $G = < a, b >$

Arithmetica Key Exchange

- Can impose other relations on $G = < a, b >$
- For example,

  $abab^{-1}a^{-1}b^{-1} = 1_G$, $a^2 = 1_G$, $b^2 = 1_G$
- Can write 1_G in infinite number of ways
- Denote this as

  $S_3 = <a,b \mid abab^{-1}a^{-1}b^{-1}, a^2, b^2>$
- A **finite presentation** of the group S_3
  - The group S_3 is a well-known symmetric group

Arithmetica Key Exchange

- Sometimes relations can be used to put any word into a **canonical** form
  - Necessary for Arithmetica
- A **subgroup** is a subset of the group that is closed under group operation
- For example
  - Integers are a subset of real numbers
  - Add two integers, you get another integer

Arithmetica Key Exchange

- Let G be a finitely presented, infinite, non-abelian group
- Alice choose subgroup

  $S_A = \langle s_0, s_1, \ldots, s_{n-1} \rangle$
- Bob chooses subgroup

  $S_B = \langle t_0, t_1, \ldots, t_{m-1} \rangle$
- Group G and subgroups S_A and S_B are **public**

Arithmetica Key Exchange

- Alice and Bob choose **private keys**

$$a = s_{\sigma(0)}^{i_0} \cdots s_{\sigma(n-1)}^{i_{n-1}} \in S_A \quad \text{and} \quad b = t_{\tau(0)}^{j_0} \cdots t_{\tau(m-1)}^{j_{m-1}} \in S_B$$

  respectively

- For **key exchange**...
  - Alice sends $\{a^{-1}t_0a, \ldots, a^{-1}t_{m-1}a\}$ to Bob
  - Bob sends $\{b^{-1}s_0b, \ldots, b^{-1}s_{n-1}b\}$ to Alice
- Rewrite to "obscure" private a and b

Arithmetica Key Exchange

- Alice can compute $b^{-1}ab$ since

$$\begin{aligned} b^{-1}ab &= b^{-1}s_{\sigma(0)}^{i_0}\cdots s_{\sigma(n-1)}^{i_{n-1}}b \\ &= b^{-1}s_{\sigma(0)}^{i_0}bb^{-1}s_{\sigma(1)}^{i_1}b\cdots b^{-1}s_{\sigma(n-1)}^{i_{n-1}}b \\ &= (b^{-1}s_{\sigma(0)}b)^{i_0}\cdots(b^{-1}s_{\sigma(n-1)}b)^{i_{n-1}}. \end{aligned}$$

- Similarly, Bob can compute $a^{-1}ba$
- Then $a^{-1}b^{-1}ab$ can be shared key
  - How can Bob compute this?

Arithmetica Example

- Let $G = \langle x,y \mid x^4, y^2, yxyx \rangle$
- Alice: $S_A = \langle s_0, s_1 \rangle = \langle x^2, y \rangle = \{1_G, x^2, y, x^2y\}$
- Bob: $S_B = \langle t_0 \rangle = \langle x \rangle = \{1_G, x, x^2, x^3\}$
- **Public**: G, S_A, S_B
- **Private**
  - Alice: $a = (x^2)^2(y)^{-1} = x^4y^{-1} = 1_Gy^{-1} = y^{-1}$
  - Bob: $b = (x)^3 = x^3$

Arithmetica Example

- Key exchange
- Alice computes: $a^{-1}t_0a = y^{-1}xy = yxy$
  - Alice sends {yxy} to Bob
- Bob computes: $b^{-1}s_0b$ and $b^{-1}s_1b$
  - Bob sends $\{x^{-2}, x^2y\}$ to Alice
- Now to establish the shared key…

Arithmetica Example

- Alice: $b^{-1}ab = (x^{-2})^2(x^2y)^{-1} = (x^2)^2(y^{-1}x^{-2})$
$= x^4y^{-1}x^{-2} = 1_G \cdot y^{-1}x^{-2} = yx^2 = x^2y.$

  then $a^{-1}(b^{-1}ab) = (y^{-1})^{-1}(x^2y) = yx^2y = x^2$

- Bob: $a^{-1}ba = (yxy)^3 = yxy \cdot yxy \cdot yxy$
$= yxy^2xy^2xy = yx \cdot 1_G \cdot x \cdot 1_G \cdot xy$
$= yx^3y = x.$

  then $a^{-1}b^{-1}a = (a^{-1}ba)^{-1} = x^{-1},$

  and, finally, $(a^{-1}b^{-1}a)b = x^{-1} \cdot x^3 = x^2.$

Arithmetica Example

- Alice and Bob shared secret: x^2
- Use this to compute symmetric key
- This example used a small, finite, non-abelian group
- In realistic implementation, G, S_A, S_B must be infinite non-abelian groups
  - Each with a large numbers of generators

Arithmetica

- Arithmetica based on a math problem known as **conjugacy problem**
- Given two words $x,y \in G$, does there exits $g \in G$ such that $y = g^{-1}xg$?
- For finitely presented group G, no efficient algorithm for this problem

Arithmetica Length Attack

- Spse, in canonical form, $w = g_0^i g_1^j g_2^k \in G$
- Define **length** of w as $|i| + |j| + |k|$
- Use this to find factors (probabilistic)
  - Existence of canonical form makes this work
  - Canonical form necessary for Arithmetica
- New attack, subject of ongoing research

Arithmetica: Bottom Line

- Relatively new, fancy mathematics
- Probably not really practical
- Shows potential for advanced math
- Not many attacks on it (yet)
- More time needed to judge security

RSA

RSA

- Invented by Cocks (GCHQ), independently, by Rivest, Shamir, Adleman (MIT)
- Let p and q be two large prime numbers
- Let N = pq be the **modulus**
- Choose e relatively prime to (p–1)(q–1)
- Find d so that ed = 1 (mod (p–1)(q–1))
- **Public key** is (N,e)
- **Private key** is d

RSA

- To encrypt M compute: $C = M^e \pmod N$
- To decrypt C compute: $M = C^d \pmod N$
- Recall that e and N are public
- If attacker can factor N, can use e to easily find d since $ed = 1 \pmod{(p-1)(q-1)}$
- Factoring the modulus breaks RSA!
- It is not known whether factoring is the only way to break RSA

Does RSA Really Work?

- Given $C = M^e \pmod N$ we must show

  $M = C^d \pmod N = M^{ed} \pmod N$

- We use **Euler's Theorem:**

  If x is relatively prime to n then $x^{\varphi(n)} = 1 \pmod n$

  - Fact: $ed = 1 \pmod{(p-1)(q-1)}$
  - Fact: $ed = k(p-1)(q-1) + 1$
  - Fact: $\varphi(N) = (p-1)(q-1)$
  - Fact: $ed - 1 = k(p-1)(q-1) = k\varphi(N)$

$$M^{ed} = M^{(ed-1)+1} = M \cdot M^{ed-1} = M \cdot M^{k\varphi(N)}$$
$$= M \cdot (M^{\varphi(N)})^k = M \cdot 1^k = M \pmod N$$

Simple RSA Example

- Example of RSA
  - Select "large" primes p = 11, q = 3
  - Then N = pq = 33 and (p–1)(q–1) = 20
  - Choose e = 3 (relatively prime to 20)
  - Find d such that ed = 1 (mod 20), we find that d = 7 works
- **Public key:** (N, e) = (33, 3)
- **Private key:** d = 7

Simple RSA Example

- **Public key:** (N, e) = (33, 3)
- **Private key:** d = 7
- Suppose message M = 8
- Ciphertext C is computed as

  $C = M^e \pmod N = 8^3 = 512 = 17 \pmod{33}$

- Decrypt C to recover message:

  $M = C^d \pmod N = 17^7 = 410{,}338{,}673$
  $= 12{,}434{,}505 * 33 + 8 = 8 \pmod{33}$

RSA Conclusions

- RSA is the "gold standard" in public key crypto
- Provides encryption and signatures
- Has stood the test of time
  - Virtually unchanged since its invention
- We look closely at RSA attacks in later section (implementation attacks)

Rabin Cipher

Rabin Cipher

- Based on difficulty of factoring
  - Like RSA
- Recall that factoring N breaks RSA
- It is not known whether factoring is the only way to break RSA algorithm
- Can be shown that breaking Rabin algorithm is equivalent to factoring

Sign and Encrypt vs Encrypt and Sign

- Before Rabin, a short detour
- Suppose we want both confidentiality and non-repudiation
- We can sign and encrypt…
- …or encrypt and sign
- Does the order matter?

Public Key Notation

- **Sign** message M with Alice's **private key**: $[M]_{Alice}$
- **Encrypt** message M with Alice's **public key**: $\{M\}_{Alice}$
- Then

$$\{[M]_{Alice}\}_{Alice} = M$$

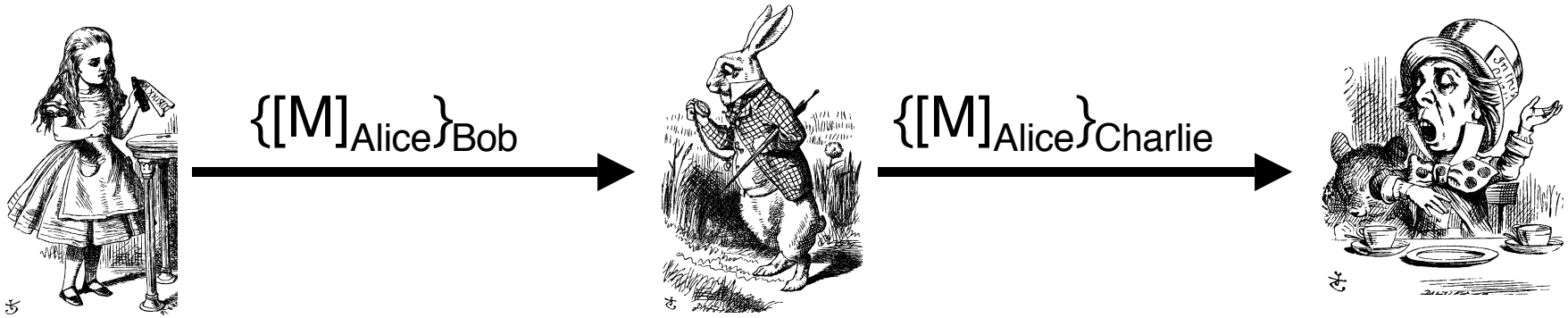
$$[\{M\}_{Alice}]_{Alice} = M$$

Confidentiality and Non-repudiation

- Suppose that we want confidentiality and non-repudiation
- Can public key crypto achieve both?
- Alice sends message to Bob
  - **Sign and encrypt** $\{[M]_{Alice}\}_{Bob}$
  - **Encrypt and sign** $[\{M\}_{Bob}]_{Alice}$
- Can the order possibly matter?

Sign and Encrypt

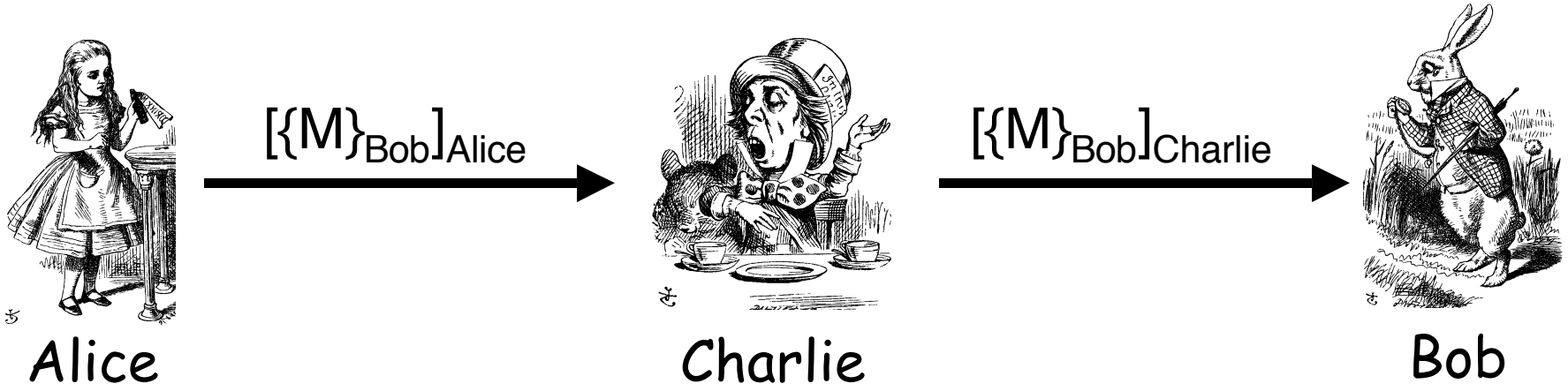
- M = "I love you"



- **Q:** What is the problem?
- ***A:*** Charlie misunderstands crypto!

Encrypt and Sign

- M = "My theory, which is mine…."



- **Note** that Charlie cannot decrypt M
- **Q:** What is the problem?
- **A:** Bob misunderstands crypto!

Rabin Cipher

- Choose N = pq, where p and q prime
- Assume p = 3 (mod 4) and q = 3 (mod 4)
  - Just to simplify discussion
- Public key: N
- Private key: (p,q)
- Encrypt: $C = M^2 \pmod N$
- Decrypt: Given p and q, we must find the square root of C, modulo N

Rabin Cipher

- How to find square root of C (mod N)?
  - Given p and q, where N = pq
- First, consider square root, mod p
- If C = 0 (mod p) then square root is 0
- If C ≠ 0 (mod p), let $y = C^{(p+1)/4} \pmod p$
- By Euler's Theorem, $C^{p-1} = 1 \pmod p$
- Therefore, $y^4 = C^{p+1} = C^2C^{p-1} = C^2 \pmod p$

Rabin Cipher

- Have $y^4 = C^{p+1} = C^2C^{p-1} = C^2 \pmod p$
  - Where y is known
- Then $y^4 - C^2 = (y^2 - C)(y^2 + C) = 0 \pmod p$
- And therefore, $y^2 = \pm C \pmod p$
- Square roots of C (mod p) are ±y or square roots of –C (mod p) are ±y
  - But not both
- Also find square root mod q and use Chinese Remainder Theorem (CRT) for result mod N

Chinese Remainder Theorem

- Use Euclidean algorithm to find r,s so that
  $qr + ps = 1$
- CRT says that x (mod pq) satisfying
  $x = a \pmod p$ and $x = b \pmod p$
  is given by $x = bpr + aqs \pmod{pq}$
- For Rabin, we have 4 cases to consider:
  $\pm a \pmod p$ and $\pm b \pmod q$

Rabin Cipher Example

- Suppose C = 16 (mod 33)
  - Have p = 3 and q = 11
- Compute $C^{(3+1)/4} = C = 16 = 1 \pmod 3$
  - Easy to verify ±1 are square roots of C (mod p)
- Compute $C^{(11+1)/4} = 5^3 = 4 \pmod{11}$
  - Easy to verify ±4 are square roots of C (mod q)
- Use CRT and consider four cases…

Rabin Cipher Example

- Euclidean algorithm: find r = –1, s = 4 gives
  $11r + 3s = 1$
- Four cases of the form
  $x = a \pmod{11}$ and $x = b \pmod{3}$, namely,
  - $x = 4 \pmod{11}$ and $x = 1 \pmod{3}$
  - $x = 4 \pmod{11}$ and $x = -1 \pmod{3}$
  - $x = -4 \pmod{11}$ and $x = 1 \pmod{3}$
  - $x = -4 \pmod{11}$ and $x = -1 \pmod{3}$
- Find $x = bpr + aqs \pmod{33}$ for each case

Rabin Cipher Example

- In this example: x = 4, 26, 7, 29
- Easy to verify $x^2 = 16 \pmod{33}$ for each case
- One of these x is the plaintext
- But which one?
  - Add header before encrypting
  - Only one x will have correct header

Chosen Ciphertext Attack

- Spse Trudy can find square roots (mod N) of C, namely, u,v, with $u \neq \pm v$
- Trudy can then factor N, since

  $u^2 = v^2 = C \pmod N$

  $u^2 - v^2 = (u - v)(u + v)$ divisible by N
- Then gcd(u + v, N) is p or q
- This breaks Rabin cipher

Chosen Ciphertext Attack

- Trudy knows M and corresponding C encrypted with Alice's public key
- Trudy gets Alice to "decrypt" C
  - That is, find square root mod N
- Suppose result of decryption is y
- If $y \neq \pm M$ then previous attack applies
  - This happens with probability 1/2
- Then Trudy can find Alice's private key

Chosen Ciphertext Attack

- Can prevent this attack by using a tricky padding scheme
- We do not discuss it here
- Mentioned in textbook
  - But not discussed in detail

NTRU Cipher

NTRU Cipher

- "Nth degree TRUncated polynomial ring" or "Number Theorists aRe Us"
  - Depending on who you ask
- Invented in 1995 by 3 mathematicians
- A complicated encryption process
  - Operations in a funny polynomial ring
- Cipher has evolved as flaws found
  - In contrast to, say, RSA
- But NTRU considered theoretically sound

NTRU

- NTRU is not widely used
- NTRU Cryptosystems, Inc.
  - Patents, challenge problems, etc., etc.
- Some standards support NTRU
- May gain more popularity
  - Unlikely to ever rival RSA
- General attack is lattice reduction

NTRU

- Three parameters: (N,p,q)
- Four sets of polynomials
  - Degree N – 1, with integer coefficients
  - Denote sets L_f, L_g, L_r, L_m
- Choose p and q so that gcd(p,q) = 1
  - Also, q > p with q "much larger" than p

NTRU Example

- All polynomials are of the form

  $a(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{N-1}x^{N-1}$

  where a_i are integers, modulo p or q
- Add polynomials in usual way
- Multiply polynomials mod x^{N-1}, that is, replace x^N with 1, x^{N+1} with x and so on
- Use symbol "$\star$" to represent this multiply

NTRU

- In math terms, NTRU polynomials in the quotient ring $R = Z[x]/(x^N - 1)$
- The messages space Lm consists of polynomials in R modulo p, that is,

$$L_m = \{M(x) \in R \mid \text{all coefficients of } M \text{ lie in } [-(p-1)/2, (p-1)/2]\}$$

NTRU

- For examples, if we choose p = 3
- Then polynomials in L_m have degree N–1 or less and coefficients in {–1,0,1}
- Let $L(d_0,d_1)$ to be polynomials in R with d_0 coeficients +1 and d_1 coeficients –1
- For example, $-1+x^2+x^3-x^5+x^9 \in L(3,2)$

NTRU

- Given NTRU parameters (N,p,q) we must select 3 more params: d_f, d_g, d
  - From NTRU recommended parameters
- Define

  $L_f = L(d_f, d_{f-1})$, $L_g = L(d_g, d_g)$ and $L_r = L(d,d)$
- Now we can (finally) generate key pair

NTRU Key Pair

- Alice selects $f(x) \in L_f$ and $g(x) \in L_g$
  - Choose f(x) invertible mod p and mod q
  - Easy to find such an f(x)
  - Let $f_p(x)$ and $f_q(x)$ be the inverses, that is,

  $f(x) \star f_p(x) = 1 \pmod p$ and $f(x) \star f_q(x) = 1 \pmod q$
- Let $h(x) = pf_q(x) \star g(x) \pmod q$
- **Public key**: h(x) and (N,p,q)
- **Private key**: $(f(x), f_p(x))$

NTRU Encryption

- Bob wants to encrypt message to Alice
- Bob select "message" $M(x) \in L_m$
- Bob choose random $r(x) \in L_r$
  - This is a "blinding" polynomial
- Using Alice's public key, Bob computes

  $C(x) = r(x) \star h(x) + M(x) \pmod q$
- The ciphertext is polynomial C(x)

NTRU Decryption

- Alice receives C(x) from Bob
- Using her private key, Alice computes

  $a(x) = f(x) \star C(x)$

  $= f(x) \star r(x) \star h(x) + f(x) \star M(x) \pmod q$
- Coefficients of a(x) taken in $-q/2$ to $q/2$
- Alice computes $b(x) = a(x) \pmod p$
- Then $M(x) = f_p(x) \star b(x) \pmod p$
- Not obvious that this works!

NTRU Example

- Suppose (N,q,p) = (11,32,3)
- And $L_f = L(4,3)$, $L_g = L(3,3)$, $L_r = L(3,3)$
- Generate key: Alice chooses f(x), g(x)
  - Both polynomials of degree 10
  - Where f(x) has 4 coefficients +1, g(x) has 3
  - Both have 3 coefficients –1
  - Both have all other coefficients 0

NTRU Example

- Suppose (N,q,p) = (11,32,3)
- And $L_f = L(4,3)$, $L_g = L(3,3)$, $L_r = L(3,3)$
- Suppose Alice chooses

$$f(x) = -1 + x + x^2 - x^4 + x^6 + x^9 - x^{10} \in L_f$$
$$g(x) = -1 + x^2 + x^3 + x^5 - x^8 - x^{10} \in L_g$$

- She computes inverse mod p and mod q

$$f_p(x) = 1 + 2x + 2x^3 + 2x^4 + x^5 + 2x^7 + x^8 + 2x^9$$
$$f_q(x) = 5 + 9x + 6x^2 + 16x^3 + 4x^4 + 15x^5 + 16x^6 + 22x^7 + 20x^8 + 18x^9 + 30x^{10}$$

NTRU Example

- Alice's private key is (f(x), $f_p(x)$)
- Alice computes

$$\begin{aligned} h(x) &= p f_q(x) \star g(x) \\ &= 8 + 25x + 22x^2 + 20x^3 + 12x^4 + 24x^5 + 15x^6 \\ &\quad + 19x^7 + 12x^8 + 19x^9 + 16x^{10} \pmod{32} \end{aligned}$$

- Alice's public key is h(x)
- Note (N,q,p) = (11,32,3) also public

NTRU Example

- Suppose Bob chooses message

$$M(x) = -1 + x^3 - x^4 - x^8 + x^9 + x^{10} \in L_m$$

- He chooses random blinding polynomial, say,

$$r(x) = -1 + x^2 + x^3 + x^4 - x^5 - x^7 \in L_r$$

- Bob computes ciphertext

$$\begin{aligned} C(x) &= r(x) \star h(x) + M(x) \\ &= 14 + 11x + 26x^2 + 24x^3 + 14x^4 + 16x^5 + 30x^6 \\ &\quad + 7x^7 + 25x^8 + 6x^9 + 19x^{10} \pmod{32} \end{aligned}$$

NTRU Example

- Alice receives C(x) and computes

$$\begin{aligned} a(x) &= f(x) \star C(x) \\ &= 3 - 7x - 10x^2 - 11x^3 + 10x^4 + 7x^5 + 6x^6 \\ &\quad + 7x^7 + 5x^8 - 3x^9 - 7x^{10} \pmod{32} \end{aligned}$$

- With coefficients between –15 and 16
- Alice reduces coefficients mod 3,

$$b(x) = -x - x^2 + x^3 + x^4 + x^5 + x^7 - x^8 - x^{10} \pmod{3}$$

NTRU Example

- Finally, Alice computes

$$f_p(x) \star b(x) = -1 + x^3 - x^4 - x^8 + x^9 + x^{10} \pmod 3$$

  which is the plaintext, M(x)
- Why does this work?
- In fact, it does not always work!
- Decryption is probabilistic…

Why Does NTRU Work?

- Ciphertext is

  $C(x) = r(x) \star h(x) + M(x) \pmod q$

- Where

  $h(x) = pf_q(x) \star g(x) \pmod q$

- To decrypt, Alice first computes

$$\begin{aligned} a(x) = f(x) \star C(x) &= f(x) \star r(x) \star h(x) + f(x) \star M(x) \pmod q \\ &= pf(x) \star r(x) \star f_q(x) \star g(x) + f(x) \star M(x) \pmod q \\ &= pr(x) \star g(x) + f(x) \star M(x) \pmod q \end{aligned}$$

Why Does NTRU Work?

- The polynomial $pr(x) \star g(x) + f(x) \star M(x)$ is probably the same mod q or not
- If so, mod q has no effect and

  $b(x) = a(x) \pmod p = f(x) \star M(x)$

  and $f_p(x) \star b(x) = M(x) \pmod p$
- But, mod q can make decryption fail!
  - Probability is low: r,g,f,M are all "small"

NTRU Lattice

- Hard math problem behind NTRU?
- Ironically, it is **lattice reduction**
  - Same problem that breaks Knapsack!
- If Trudy can determine f(x) or $f_q(x)$, from h(x), she gets Alice's private key
- Recall $h(x) = pf_q(x) \star g(x) \pmod q$
- Equivalently, $h(x) \star f(x) = pg(x) \pmod q$

NTRU Lattice

- Denote $h(x) = h_0 + h_1x + \ldots + h_{N-1}x^{N-1}$
- Define

$$H = \begin{bmatrix} h_0 & h_{N-1} & h_{N-2} & \cdots & h_1 \\ h_1 & h_0 & h_{N-1} & \cdots & h_2 \\ \vdots & & \ddots & & \vdots \\ h_{N-1} & h_{N-2} & h_{N-3} & \cdots & h_0 \end{bmatrix}$$

- Let h be coefs of h(x), as a column vector and similarly for f(x) and g(x)

NTRU Lattice

- By the definition of "$\star$", we have

  Hf = pg (mod q)

- Equivalent to block matrix equation

$$MV = \begin{bmatrix} I_{N\times N} & 0_{N\times N} \\ H_{N\times N} & qI_{N\times N} \end{bmatrix} \begin{bmatrix} f \\ s \end{bmatrix} = \begin{bmatrix} f \\ pg \end{bmatrix} = W \pmod{q}$$

- That is, f = f and Hf + qs = pg (mod q)

NTRU Lattice

- Trudy gets private key if she gets V or W
  - W in lattice spanned by columns of M
  - W has special form (number of +1 and –1)
  - W is a "short" vector
- Lattice reduction attack!
  - Just like the knapsack?
- No, this NTRU lattice is hard to break!
  - As far as anybody knows…

NTRU Lattice

- Note that success against this NTRU lattice would recover **private key**
- Knapsack lattice just broke 1 message
- Unfair to compare these attacks?
- We can rewrite NTRU attack so it breaks only a single message
- And it's still a hard problem!

Why Bother with NTRU?

- Efficiency — for public key, NTRU is fast!
- Compared to RSA 512-bit modulus, NTRU inventors claim for "equivalent" NTRU
  - Encryption is 5.9 times faster
  - Decryption is 14.4 times faster
  - Key creation is 5.0 times faster
- Good for resource constrained environment?
- But, the higher the security level, the **less** impressive the advantage for NTRU

NTRU Attacks

- Lattice reduction
  - Generic attack (like factoring for RSA)
- Meet-in-the-middle
  - Square root of "exhaustive search" work
  - Inherent in use of polynomials
- Multiple transmission
  - Encrypt M(x) multiple times with different r(x)
  - Complex padding can prevent it
- Chosen ciphertext
  - Broke earlier version of NTRU

NTRU Conclusions

- A very different public key system
- Based on "hard" lattice problem
- Has evolved since its introduction
- Considered theoretically sound
- Not widely used
- An interesting system

ElGamal Signature

ElGamal Signature

- Based on discrete log problem
  - Same hard problem as Diffie-Hellman
- Only for signatures
  - No encryption
- Widely used in the form of the Digital Signature Standard (DSS)

ElGamal

- Alice choose large prime p and number s and a, both between 2 and p – 2
- Alice computes $\alpha = s^a \pmod p$
- **Private**: a **Public**: (p,s,α)
- Spse Alice wants to sign M
  - Selects random k with gcd(k, p – 1) = 1, computes $r = s^k \pmod p$ and $t = k^{-1}(M - ra) \pmod{(p-1)}$
- Alice sends the triple (M, r, t)

ElGamal

- **Private**: a **Public**: (p,s,α)
- Where $\alpha = s^a \pmod p$
- Alice sends the triple (M, r, t), where

  $r = s^k \pmod p$ and $t = k^{-1}(M - ra) \pmod{(p-1)}$
- To verify signature, Bob computes

  $v = s^M \pmod p$ and $w = \alpha^r r^t \pmod p$
- If $v = w \pmod p$ the signature is accepted

ElGamal

- Why does this work?

$$\begin{aligned} w = \alpha^r \cdot r^t \pmod{p} &= (s^a)^r (s^k)^t \pmod{p} \\ &= s^{ra} \cdot s^{kk^{-1}(M-ra)} \pmod{p} \\ &= s^M \pmod{p} \\ &= v \pmod{p}. \end{aligned}$$

- If Trudy can compute discrete logs, she can find private key a from α
- To forge signature, Trudy must find r,t so that $s^M = \alpha^r r^t$
- Unknown whether this is equivalent to discrete log problem

ElGamal Issues

- If all prime factors of p – 1 are small, easy to compute discrete log
- If Trudy can guess k, she can find private key (with high probability)
- If Alice repeats k, Trudy can find Alice's private key
- Alice must sign h(M), not M, or else Trudy can forge Alice's signature
  - But "message" M is nonsense

Public Key Systems

- A quick intro to several systems
- Public key encryption/decryption
  - RSA, Rabin, NTRU, Knapsack
- Key exchange protocols
  - Diffie-Hellman, Arithmetica
- Signature scheme
  - ElGamal

Public Key Systems

- Each rests on a (presumed) difficult math problem
- RSA, Rabin
  - Factoring
- Diffie-Hellman, ElGamal
  - Discrete log
- Lack of "genetic diversity" in public key

Public Key Systems

- Next, we discuss factoring algorithms
- Then discrete log algorithms
- Finally, we consider implementation attacks on RSA
  - Do not attack algorithm directly
  - Attack based on timing the computation
  - Attack based on induced error