Lattice-based Encryption

Phong Nguyễn

October 2020

Today

- Lattice Analogues of:
  - RSA: Encryption with Trapdoors
  - Diffie-Hellman
  - El Gamal: Encryption without Trapdoors

Lattice Cryptography: Design

Lattice-based Crypto

- Two Types of Techniques
  - Cryptography using trapdoors, i.e. secret short basis of a lattice. Similarities with RSA/Rabin cryptography.
  - Cryptography without trapdoors. Similarities with DL cryptography.
- Case study: Encryption.

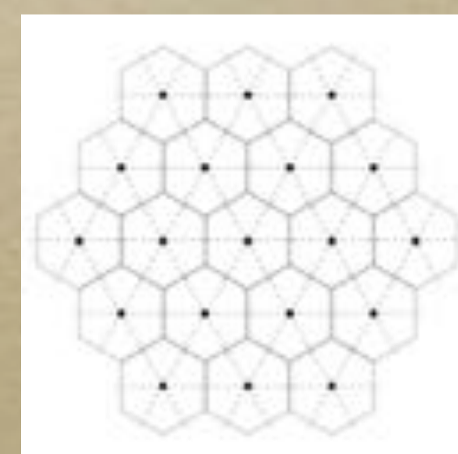
Trapdoor-based Encryption: GGH and NTRU

Remember

- $N=pq$ product of two large random primes.
- $ed \equiv 1 \pmod{\phi(N)}$ where $\phi(N)=(p-1)(q-1)$.
  - e is the public exponent
  - d is the secret exponent
- Then $m \to m^e$ is a trapdoor one-way permutation over $\mathbf{Z}/N\mathbf{Z}$, whose inverse is $c \to c^d$.

Bounded Distance Decoding (BDD)

- Input: a basis of a lattice L of dim d, and a target vector t very close to L.
- Output: v∈L minimizing ||v-t||. Easy if one knows a nearly-orthogonal basis.



Reducing Modulo a Lattice

- If L is an integer lattice, the quotient $\mathbf{Z}^n/L$ is a finite group, with many representations: lattice crypto works modulo a lattice.
- We call L-reduction any efficiently computable map f from $\mathbf{Z}^n$ s.t. $f(x)=f(y)$ iff $x-y \in L$.

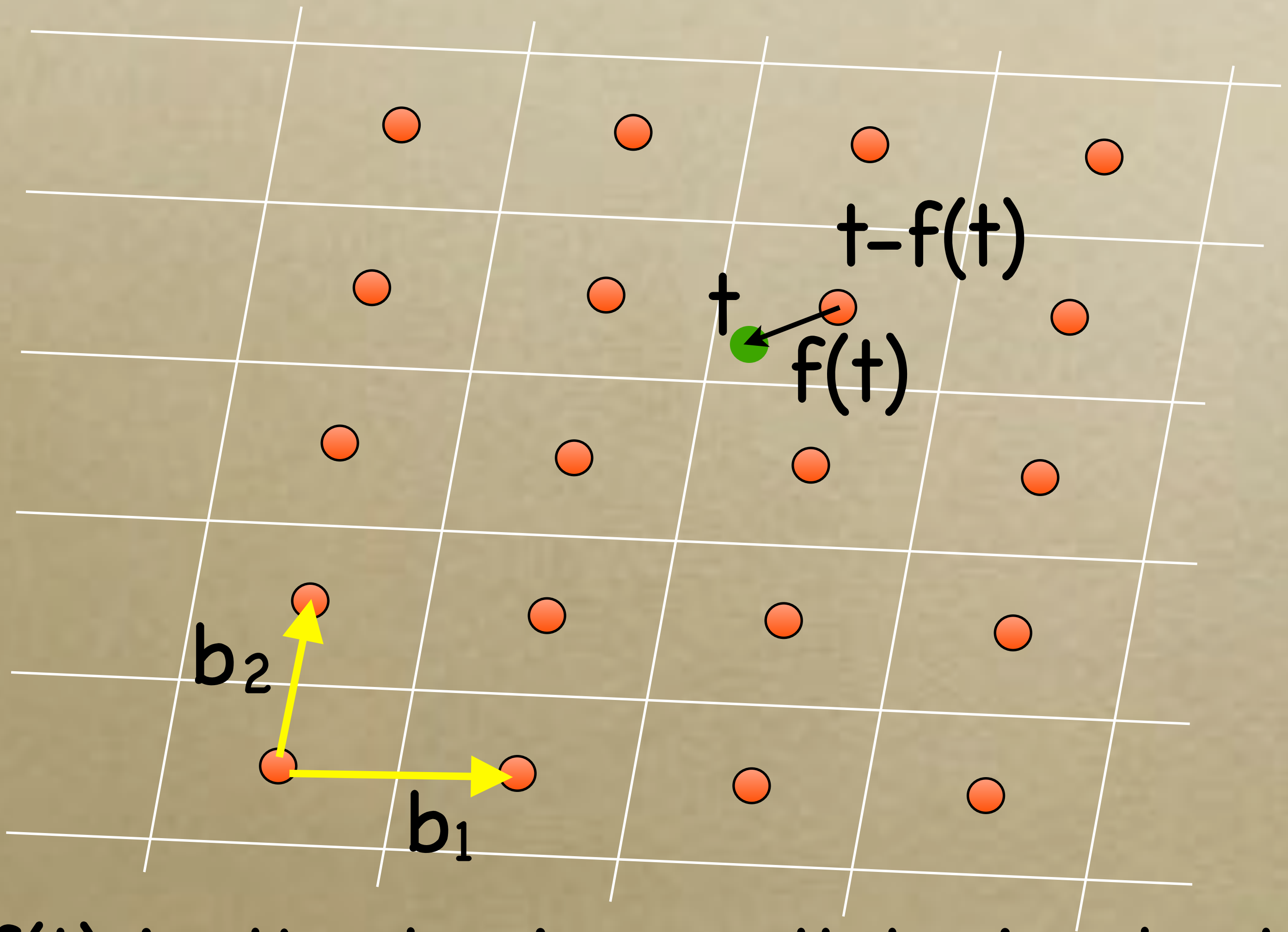
One-Way Functions from BDD

- If BDD is hard over a ball, any public L-reduction f is a one-way function over the same ball.
  - Let (t,L) be a BDD instance: t=v+e where v∈L and e is very short.
  - Then f(t)=f(e) because t-e=v∈L: if f is not one-way, then given f(e), one can recover e and also the BDD solution v=t-e.

Building L-Reductions

- Any basis provides two L-reductions, thanks to Babai's nearest plane algorithm and rounding-off algorithm.
- NTRU encryption implicitly uses a L-reduction.

Ex: Babai's rounding off



Choose f(t) in the basis parallelepiped s.t. t-f(t)∈L

Ex: Babai's rounding off

- Let t in $\mathbf{Z}^n$.
- Let B the lattice basis.
- Solve $t=uB$ where u in $\mathbf{Q}^n$.
- Return $f(t)=(u-\lfloor u \rceil)B$

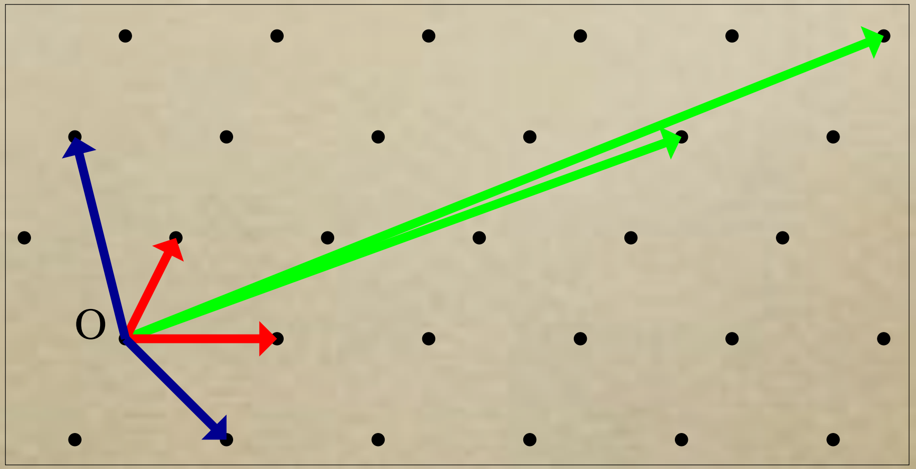
Ex: Babai's nearest plane algorithm

- Let t in $\mathbf{Z}^n$.
- Let B the lattice basis and B* its Gram-Schmidt orthoganlization.
- Find v=uB where u in $\mathbf{Z}^n$ s.t. t-v = xB* where each coordinate of x is $\leq$ 1/2 in absolute valute
- Return f(t)=t-v.

Solving BDD by L-reduction

- The L-reductions derived from Babai's algorithms leave some set invariant: there exists $D(B) \subseteq \mathbf{Z}^n$ s.t. $f(x)=x$ for all $x \in D(B)$. This allows to solve BDD when the error $\in D(B)$.
- The largest ball inside $D(B)$ depends on the quality of the basis.

Deterministic Public-Key Encryption [GGH97-Micc01]



- Secret key = Good basis
- Public key = Bad basis
- Message = Short vector
- Encryption = L-reduction with the public key
- Decryption = L-reduction with the secret key
- Optimization: Ntrū

Encryption with the Hardest Lattices

SIS Trapdoors

- [Ajtai1999,AlwenPeikert2010,MicciancioeikErt2012] showed that it is possible to generate $g_1,\ldots,g_m \in (\mathbf{Z}/q)^n$ with distribution statistically close to uniform, together with a short basis of the SIS lattice $L=\{x=(x_1,\ldots,x_m)\in\mathbf{Z}^m$ s.t. $\sum_i x_i g_i = 0\}$.

Optimizing Encryption: NTRU

Optimization: NTRU Encryption

- Ring $R=\mathbf{Z}[X]/(X^N-1)$, secret key $(f,g)\in R^2$, public key $h=g/f \pmod q$.
- Encryption can be viewed [Mi01] as L-reducing a short vector with the Hermite normal form, where $L=\{(u,v)\in R^2 \text{ s.t. } u\equiv pv*h \pmod q\}$.
- Decryption is a special BDD algorithm using the secret key (f,g).

NTRU Encryption

- Invented by Hoffstein, Pipher and Silverman in 1996 (CRYPTO rump session):
  - First published in 1998
  - First cryptanalysis (Coppersmith-Shamir) in 1997!
  - Perhaps the fastest public-key encryption scheme known, and one of the most studied.

Key Generation

- Let N be a prime number, e.g. 251
- Consider the ring $R=\mathbf{Z}[X]/(X^N-1)$
- Let p and q be two small coprime integers:
  - p=3 and q a small power of 2 (128 or 256)
  - p=2 and q a small prime number

Key Generation

- The secret key is two polynomials f and g in R with very small coefficients:
  - f and g could be ternary (0, 1, -1) or binary (0, 1).
  - f must be invertible mod q and p. Let f_p and f_q be the inverse.
- The public key is $h=g*f_q$ mod q so $h*f = g$ mod q.

Encryption

- To encrypt a message m (a polynomial in R having small coefficients):
  - Choose at random a sparse polynomial r in R with very small coefficients.
  - The ciphertext is $c = m + p\ r*h \bmod q$.
  - Encryption is probabilistic.

Decryption

- Multiplying by the secret key f, we can get:
  - c*f = m*f + p r*g (mod q).
- If we could get the exact value of m*f + p r*g over the integers, we could easily recover m mod p.
- Note: both products m*f and r*g involve only polynomials with small coefficients, possibly sparse.

Products of Small Polynomials

- Let f and g be two polynomials in R such that
  - f only has 0,1-coefficients.
  - g has small coefficients with identical distribution.
- Then any coeff of f*g is just a sum of coeffs of g: the distribution should approximately be Gaussian with small standard deviation.

Impact on Decryption

- This means that the coefficients of both m*f and r*g lie in a short interval, so that the coefficients of m*f + p r*g lie in an interval of length possibly <= q.
- Then, one could recover the exact value of m*f + pr*g from its value mod q.

Efficiency of Encryption

- One needs to compute p r*h mod q, where h has mod q coefficients and r is sparse with coefficients 0,+1,-1: each coefficient of p r*h is just a sum/difference of coefficients of p*h.
- Overall, this is $O(N^2)$ additions mod q, possibly less since r is "sparse".

Efficiency of Decryption

- The computation of c*f mod q: again, f only has 0,+1,-1 coefficients. This is $O(N^2)$ additions mod q.
- Multiplication by the inverse of f mod p.
  - If we choose a special form for f, this can be negligible.
  - Otherwise, it is $O(N^2)$ mults mod p.

Security

- The main security parameter is N, but other parameters are important.
- Key-recovery attacks
  - Brute force over f and g.
  - Square-root attack by Odlyzko.
  - Lattice attack by [CoppersmithShamir1997]. NTRU claims that this attack takes exponential time.

Lattice Attack on NTRU

- The equation $h*f = g \bmod q$ can be interpreted in terms of lattice.
- The set L of all polynomials u and v in R such that $h*u = v \bmod q$ is a lattice of $\mathbf{Z}^{2N}$, of dimension 2N.
- The pair (f,g) belongs to the lattice L and it is very short because f and g have small coefficients: its norm is $O(N^{1/2})$.

Lattice Interpretation of NTRU Encryption

- The encryption equation $c = m + p\, r * h \pmod q$ means that the vector $(0,c)$ in $\mathbf{Z}^{2N}$ is close to the lattice vector $(pr, pr*h \bmod q)$ in L, because the difference is (pr, m).
- This is a BDD problem like in GGH encryption.

Trapdoor-less Encryption

Diffie-Hellman Key Exchange

- Let $G = \langle g \rangle$ be generated by g of order q.

Alice

$a \in_R \mathbf{Z}/q\mathbf{Z}$ $\xrightarrow{g^a \in G}$

Bob

$\xleftarrow{g^b \in G}$ $b \in_R \mathbf{Z}/q\mathbf{Z}$

- Both can compute the shared key g^{ab}.
- This key exchange is the core of El Gamal public-key encryption.

El Gamal Encryption

- Let G be a cyclic group $\langle g \rangle$ of order q.
- Secret key $x \in_R \mathbf{Z}/q\mathbf{Z}$. Public key $y = g^x \in G$.
- Encrypt $m \in G$ as $(a,b) \in G^2$.
  - $a = g^k \in G$ where $k \in_R \mathbf{Z}/q\mathbf{Z}$
  - $b = m y^k \in G$
- Decrypt (a,b) by recovering $y^k = g^{kx} = a^x$.

El Gamal Encryption

- Behind El Gamal, there is the Diffie-Hellman key exchange.
  - Alice has a secret key $x \in_R \mathbf{Z}/q\mathbf{Z}$ and discloses $y=g^x \in G$
  - Bob selects a one-time key $k \in_R \mathbf{Z}/q\mathbf{Z}$ and discloses $g^k \in G$
  - Both can compute the shared key g^{kx}.

Abstracting DH

- Let $e: \mathbf{Z}_q \times \mathbf{Z}_q \to G$, $(a,b) \mapsto g^{ab}$. This map is a pairing: it is bilinear.
- Let $f: \mathbf{Z}_q \to G$, $a \mapsto g^a$ be the DL one-way function
- $e(a,b)$ can be computed using $(f(a),b)$ or $(a,f(b))$, i.e. even if a or b is hidden by f.
- Security = hard to distinguish $(f(a),f(b),e(a,b))$ from $(f(a),f(b),\text{random})$. This is called DDH.

DH with Lattices?

- What would be the pairing?
- What would be the one-way function to hide inputs?

The SIS One-Way Function

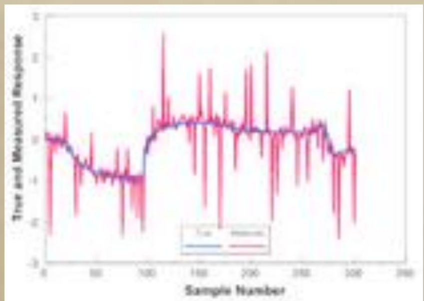
- Let $g_1,\ldots,g_m$ be uniformly distributed over G.
- The input set is $\{-1,0,1\}^m$ or any small subset of $\mathbf{Z}^m$.
- $f_g(x_1,\ldots,x_m) = \sum_i x_i g_i \in G$.
- Inversion is as hard as SIS.

The LWE One-Way Function

- Let $g_1,\dots,g_m$ be uniformly distributed over G.
- The input is a pair (s,e) where s is a character in G^x and e is small $\in(\mathbf{R}/\mathbf{Z})^m$
- Then $f^x_g(s,e)= (s(g_1),\dots,s(g_m))+e \in(\mathbf{R}/\mathbf{Z})^m)^m$
- Inversion is LWE.

Pairing from Lattices

- Let $g_1,\ldots,g_m$ in G. The dual group $G^\times$ induces a pairing $G^\times \times \mathbf{Z}^m \rightarrow \mathbf{R}/\mathbf{Z}$ by $\varepsilon(s,(x_1,\ldots,x_m)) = s(\Sigma_i\, x_i\, g_i)$
- Let $y=f_g(x_1,\ldots,x_m)=\Sigma_i\, x_i\, g_i \in G$ where x_i's small. and $b=f^\times_g(s,e)= (s(g_1),\ldots,s(g_m))+e \in(\mathbf{R}/\mathbf{Z})^m$, e small.
- Then $\varepsilon(s,(x_1,\ldots,x_m))$ can be computed from (s,y) or $(b,(x_1,\ldots,x_m))$ as $s(\Sigma_i\, x_i\, g_i) = \Sigma_i\, x_i\, s(g_i) \approx \langle(x_1,\ldots,x_m),b\rangle$ because the x_i's are small.



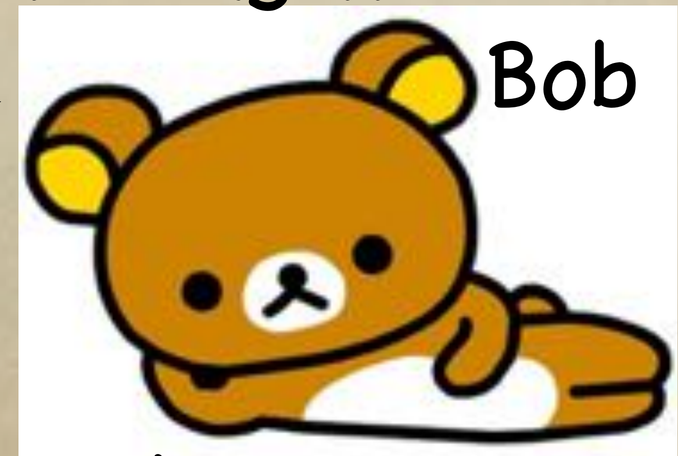
Noisy Key-Exchange from Lattices

- Let $g_1,\ldots,g_m$ generate G.

$b=f^{\times}_{g}(s,e)=(s(g_1),\ldots,s(g_m))+e$

Alice

$s \in_R G^{\times}$



Bob

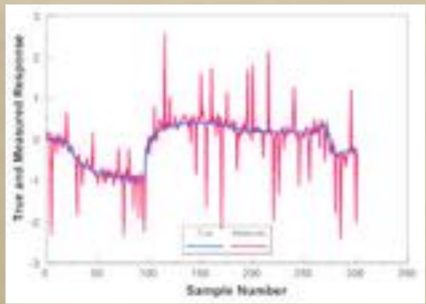
$y=f_g(x_1,\ldots,x_m)=\sum_i x_i g_i$

short $(x_1,\ldots,x_m)$

- Both compute an approx of $\varepsilon(s,(x_1,\ldots,x_m))=s(y)$:

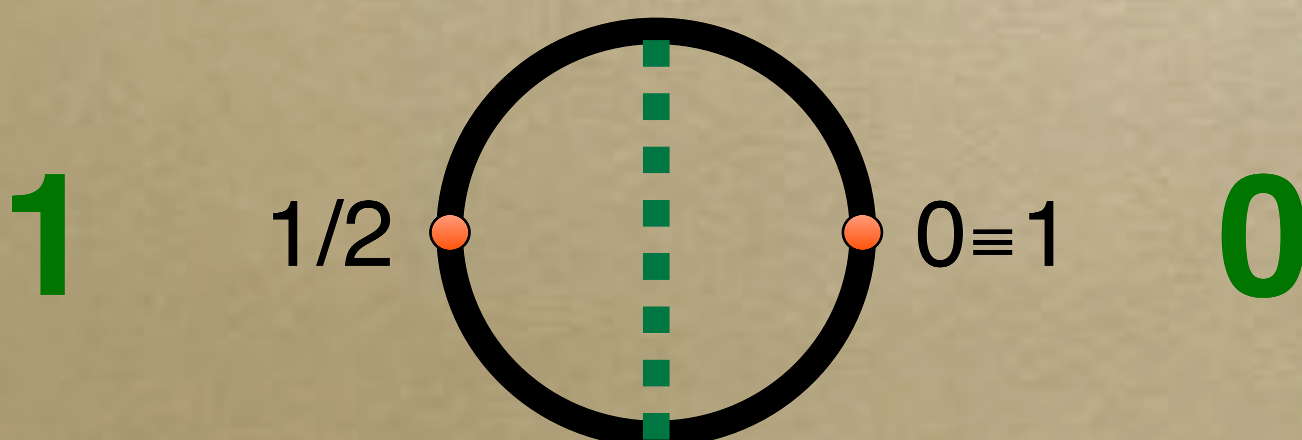
  Alice computes $s(y)+e'$ and

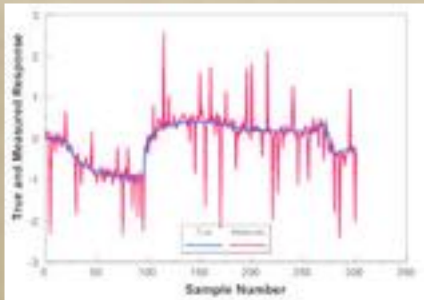
  Bob computes $\sum_i x_i b_i$.



≠ Diffie-Hellman: The Noise

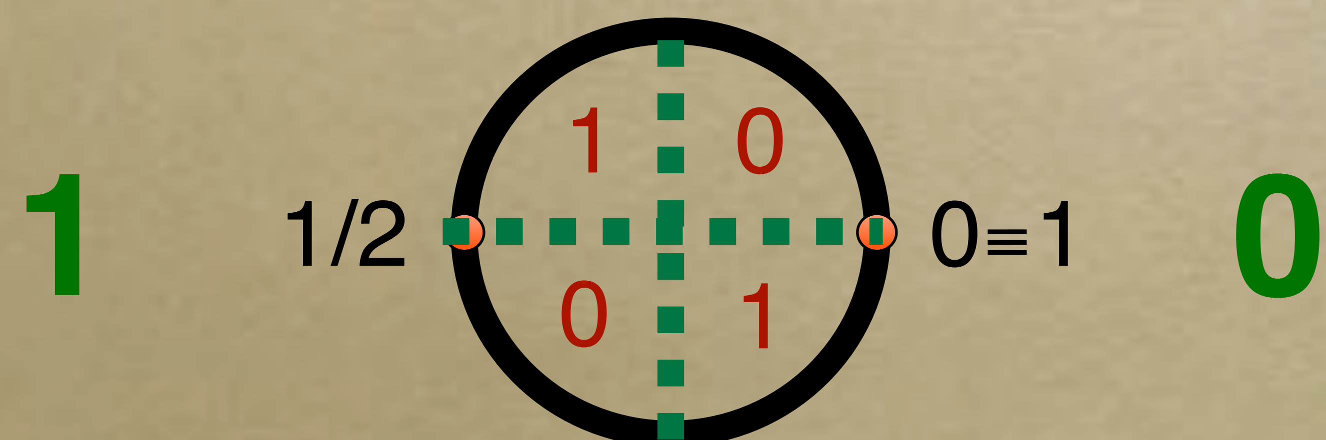
- The two values computed by Alice and Bob are elements of the torus (**R/Z**) which are **close** to each other.
- But how can they extract a bit?

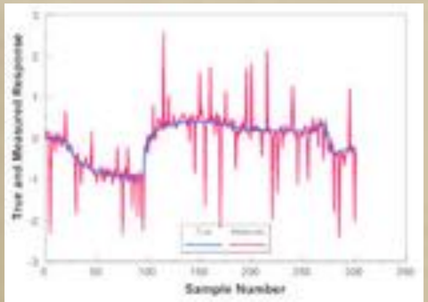




Key Reconciliation

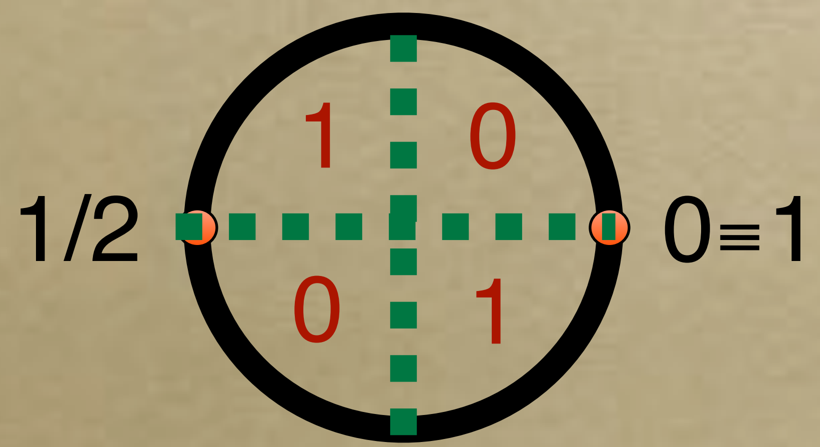
- If Alice's approximation is $k \in (\mathbf{R}/\mathbf{Z})$, Alice agrees on the bit $1- \lfloor 2(k-1/2) \rceil$ and sends the quadrant-bit to help Bob correct his approximation: this bit is uniformly distributed.

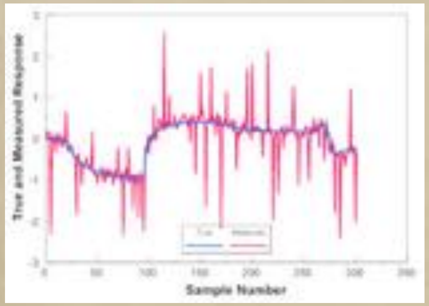




Key Reconciliation

- More sophisticated key reconciliation are possible using higher-dimensional lattices: see NewHope and other NIST submissions.





El Gamal Encryption from Lattices

- This key exchange gives rise to two El Gamal-like public-key encryption schemes, because the lattice pairing is not symmetric.
- These El-Gamal-like schemes are IND-CPA-secure under the hardness of LWE/SIS.
- Similarly, many LWE/SIS schemes can be viewed as analogues of the RSA/DL world.

Lattice El Gamal I [Regev05]

- Let $g_1,\ldots,g_m$ generate G.
- Secret key $s \in_R G^\times$.

  Public key $b = f^\times_g(s,e) = (s(g_1),\ldots,s(g_m)) + e$.
- Encrypt $m \in \{0,1\}$ as $(y,c) \in G \times (\mathbf{R}/\mathbf{Z})$
  - $y = f_g(x_1,\ldots,x_m) = \sum_i x_i g_i$ where $(x_1,\ldots,x_m)$ is short
  - $c = \sum_i x_i b_i + (m/2)$
  - Decrypt (y,c) as $\lfloor 2(s(y)-c) \rceil \in \{0,1\}$

Lattice El Gamal II [GPV08]

- Let $g_1,\ldots,g_m$ generate G.
- Secret key: short $(x_1,\ldots,x_m)\in\mathbf{Z}^m$.
  Public key: $y = f_g(x_1,\ldots,x_m) = \Sigma_i\, x_i\, g_i$
- Encrypt $m\in\{0,1\}$ as $(b,c)\in(\mathbf{R}/\mathbf{Z})^m \mathrm{x} (\mathbf{R}/\mathbf{Z})$
  - $b=f^x{}_g(s,e)= (s(g_1),\ldots,s(g_m))+e$ where $s\in_R G^x$
  - $c = s(y) + e' + (m/2)$
  - Decrypt (b,c) as $\lfloor 2(\Sigma_i\, x_i\, b_i - c) \rceil \in\{0,1\}$

Homomorphic Encryption

- El Gamal is well-known to be homomorphic with respect to the group operation G: the product of ciphertexts is a ciphertext of the product.
- Our Lattice El Gamal are bounded-homomorphic.
- How about our Trapdoor Encryption?