LATTICE-BASED ENCRYPTION

PHONG NGUYEN

http://www.di.ens.fr/~pnguyen

October 2024

4 NIST STANDARDS

- KEM/encryption (1):
  - ➤ **Crystals-Kyber**: El Gamal using MLWE lattices
- Signatures (3)
  - ➤ **Crystals-Dilithium**: Schnorr/Fiat-Shamir with aborts using MLWE
  - ➤ **Falcon**: Rabin/GGH-GPV using NTRU lattices
  - ➤ **Sphincs+**: Hash-based

TODAY

Lattice Analogues of:

- RSA
    - ➤ Encryption with Trapdoors
- Diffie-Hellman and El Gamal
    - ➤ Encryption without Trapdoors

LATTICE CRYPTO DESIGN

2 types of Lattice techniques:

- Cryptography with trapdoors

  i.e. secret short basis of a lattice

  - ➤ Similarities with RSA/Rabin cryptography

- Cryptography without trapdoors

  - ➤ Similarities with Discrete logarithm (DL) cryptography

- Case study: Encryption

RSA

Encryption with Trapdoors

TRAPDOOR-BASED ENCRYPTION: GGH AND NTRU

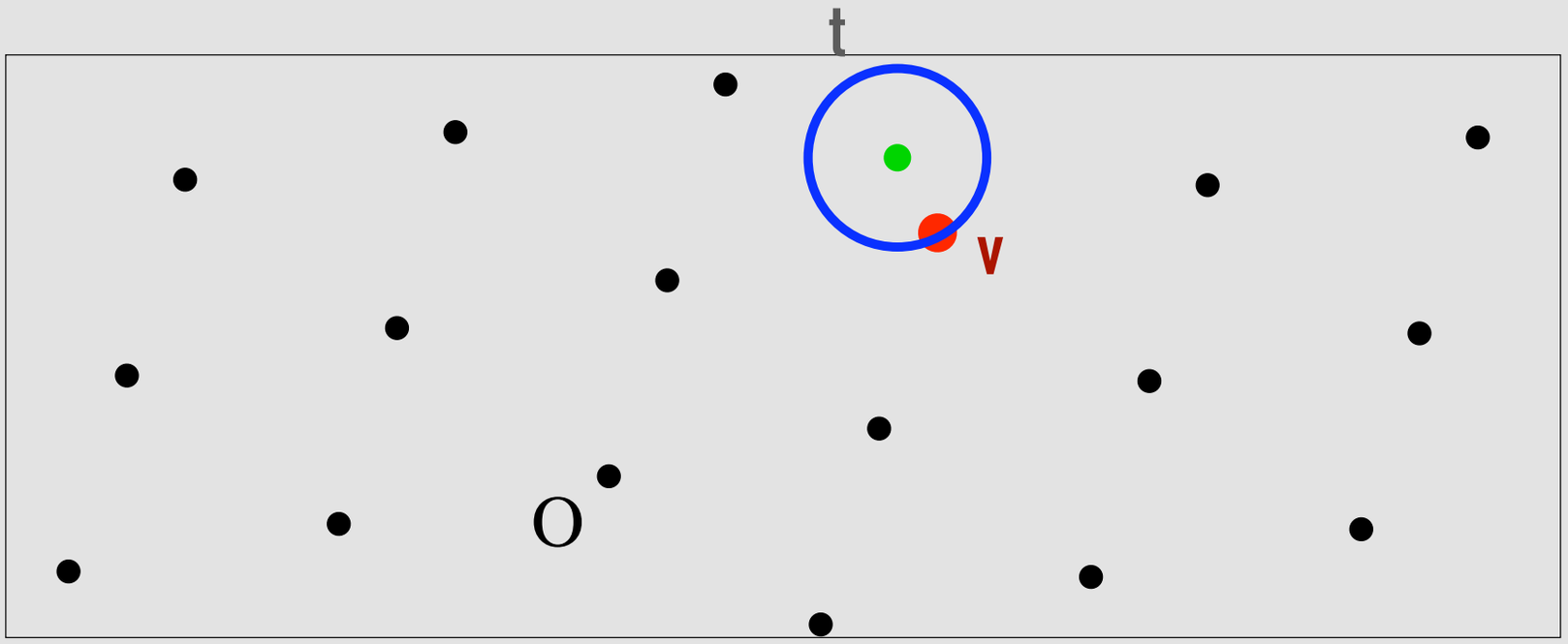
REMEMBER

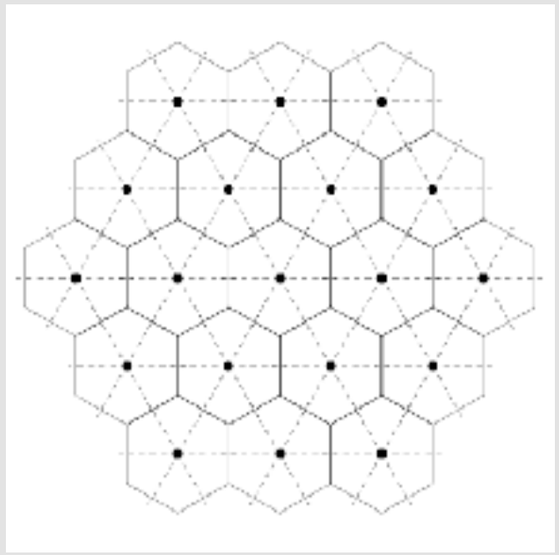
- N=pq product of two large random primes.

- ed≡1 (mod ϕ(N)) where ϕ(N)=(p-1)(q-1)
  - e is the public exponent
  - d is the secret exponent

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 $f : \mathbf{Z}^n \to \mathbf{Z}^n$ s.t. $f(x)=f(y)$ iff $x-y \in L$.

ONE-WAY FUNCTIONS FROM BDD

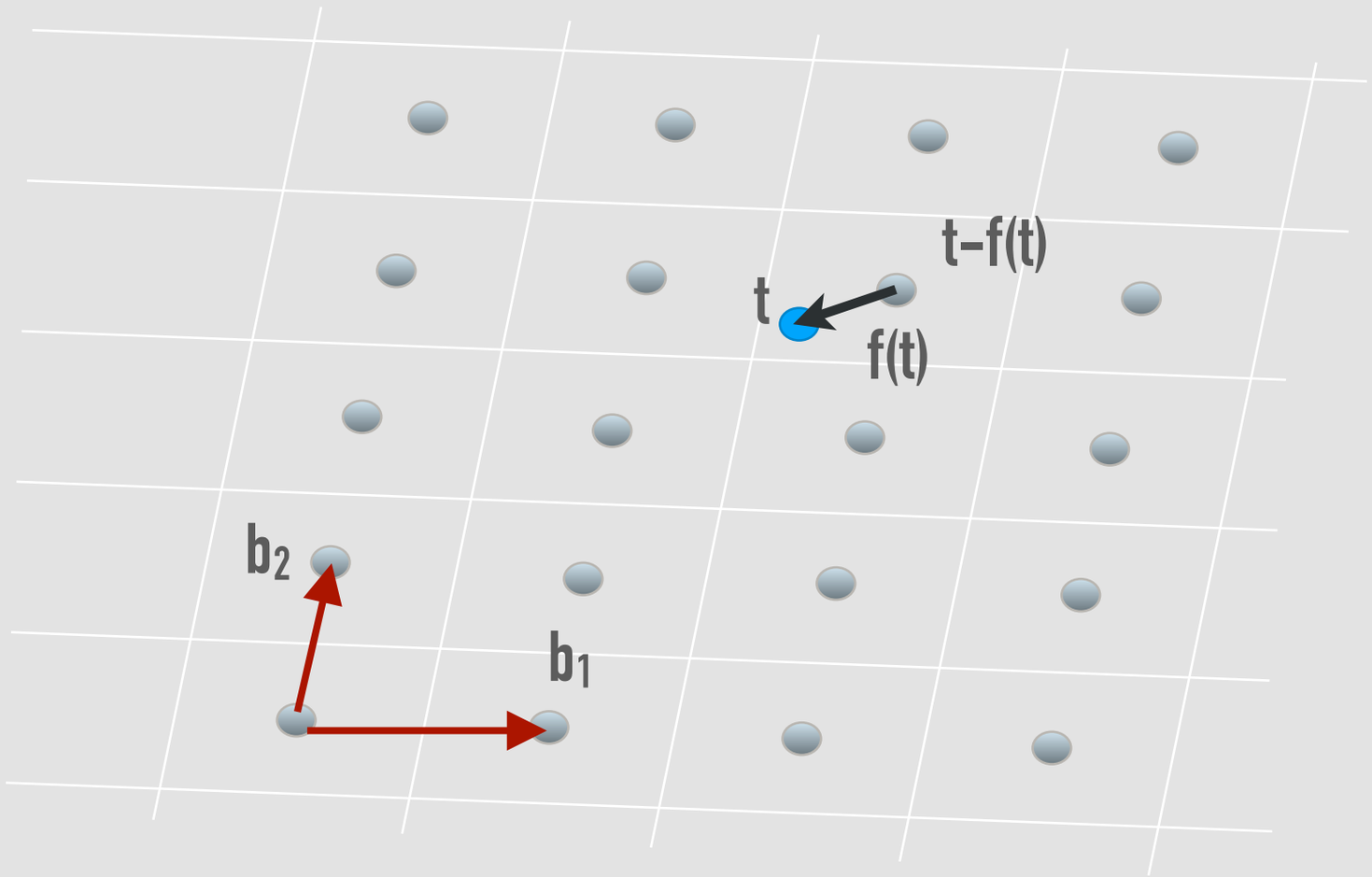
- If BDD is hard over a ball, any public L-reduction f is a one-way function over the same ball.
- Let (t,L) be a BDD instance: t=v+e where v∈L and e is very short.
- Then f(t)=f(e) because t-e=v∈L: if f is not one-way, then given f(e), one can recover small e' s.t. f(e)=f(e'), therefore e-e' ∈L. If e and e' are sufficiently small, then $\|\vec{e} - \overrightarrow{e'}\| \leq \lambda_1(L)$ so e=e'. One recovers 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.

EXAMPLE: BABAI'S ROUNDING OFF

- Any basis provides two L-reductions, thanks to Babai's nearest plane algorithm and rounding-off algorithm.



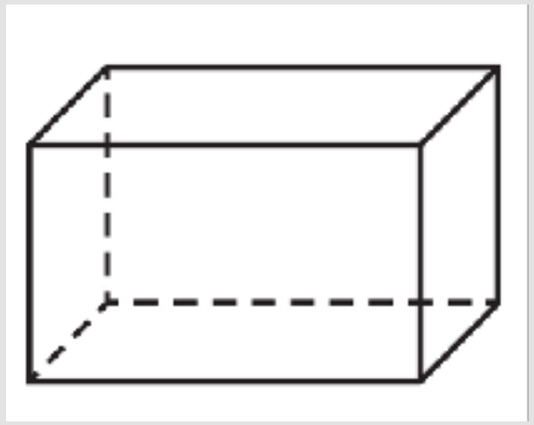
- Choose f(t) in the basis parallelepiped s.t. t-f(t)∈L

EXAMPLE: BABAI'S ROUNDING OFF

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

EXAMPLE: 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 ≤ 1/2 in absolute value
    - For i=n downto 1
        - Compute $\mu_i = \dfrac{\langle \vec{t}, \overrightarrow{b_i}^{\star} \rangle}{\| \overrightarrow{b_i}^{\star} \|^2}$
        - $\vec{t} \leftarrow \vec{t} - \lfloor \mu_i \rceil \vec{b}_i$
    - Return $\vec{v} = \sum_{i=1}^{n} \lfloor \mu_i \rceil \vec{b}_i$ and $\vec{u} = (\mu_1, \ldots, \mu_n)$
- 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 $x \in D(B)$.

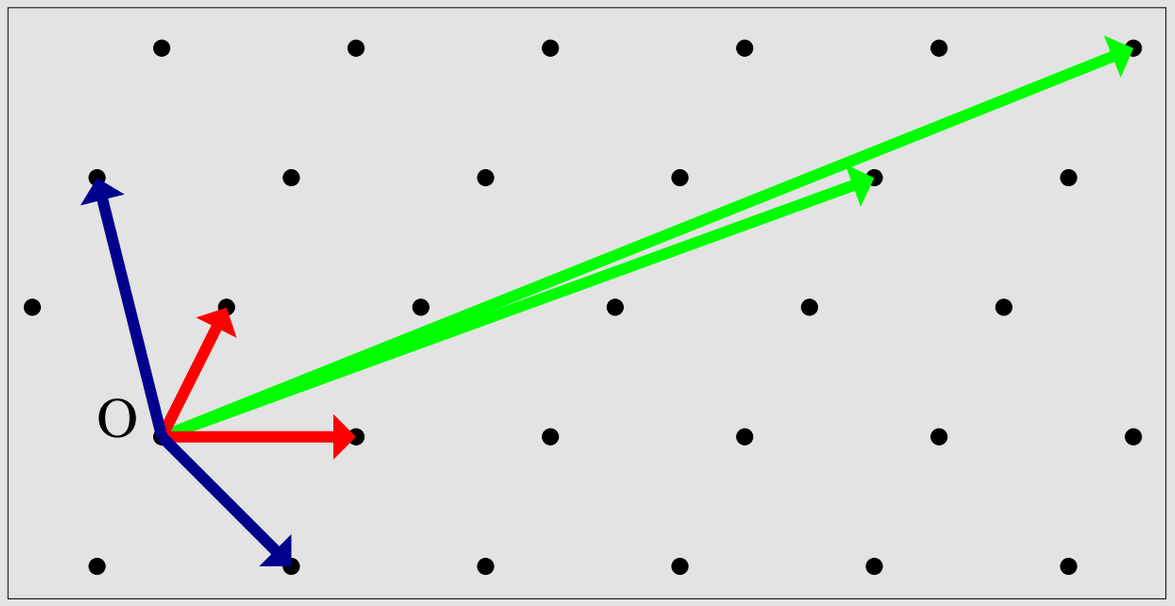
➤ This allows to solve BDD when the error$\in D(B)$.

- 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 like $\{-1,0,1\}^n$.
- **Encryption** = L-reduction with the public key
- **Decryption** = L-reduction with the secret key

➤ Optimization:



CHOOSING A BAD BASIS

- Intuitively, most bases are « bad ».

- There is a canonical basis for integer lattices: the Hermite normal form.

  ➤ If L is full-rank, the HNF is a lower-triangular matrix with positive diagonal dominating each column.

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 $L=\{x=(x_1,\ldots,x_m)\in\mathbf{Z}^m \text{ s.t. } \Sigma_i x_i g_i = 0\}$.

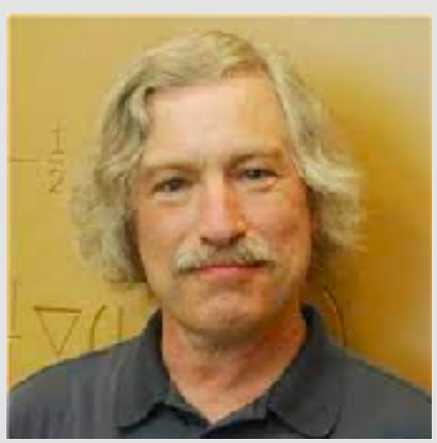
OPTIMIZING ENCRYPTION: NTRU

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 $L=\{(u,v)\in R^2$ 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 (ANTS conference)
- First cryptanalysis (Coppersmith-Shamir) in 1997!
- One of the fastest public-key encryption known, and one of the most studied.
- Based on polynomials modulo a small integer.

REFERENCES

- Hoffstein, Pipher, Silverman: **NTRU: A Ring-Based Public Key Cryptosystem**. ANTS 1998: 267-288
- https://ntru.org/
- Submission to NIST
- Reference implementations in C

PARAMETERS

- 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:
  - In 1998:

N	p	q
107	3	64
167	3	128
503	3	256

  - In 2020:

N	p	q
509	3	2048
677	3	2048
821	3	4096

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 $h = g * f_q \bmod q$ so $h * f = g \bmod q$.

CHOICE OF F AND G

- In 1998: f has d_f coeffs $+1$ and d_f-1 coeffs-1,

g has d_g coeffs $+1$ and d_g coeffs -1

N	d_f	d_g
107	15	12
167	61	20
503	216	72

- In 2020: f is uniform $\{0, \pm 1\}$

g has d_g coeffs $+1$ and d_g coeffs -1

N	d_g	q
509	127	2048
677	127	2048
821	255	4096

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 mod q.

➤ Encryption is probabilistic.

EFFICIENCY OF ENCRYPTION

- We can store H=p*h mod q instead of h, so that encryption is simply c = m + r*H mod q.
- The product r*H is special cause has 0,±1 coefficients. Each coefficient of r*H can be computed with at most N additions and subtractions, rather than O(N) multiplications mod q.
- So encryption costs at most N^2 additions mod q, even less if r is sparse.

DECRYPTION

- Multiplying by the secret key f, we can get:

➤ $c*f = m*f + p\ r*g \pmod q$

- 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 coefficient of f*g is just a sum of coeffs of g: the distribution should approximately be Gaussian with small standard deviation.

HOW SMALL ARE THE COEFFICIENTS?

- In NTRU-2020
  - m has d_g coeffs +1 and d_g coeffs -1, so each coefficient of m*f is in $\{-2d_g,\ldots,2\ d_g\}$
  - g has d_g coeffs +1 and d_g coeffs -1,so each coefficient of r*g is in $\{-2d_g,\ldots,2\ d_g\}$
  - So each coefficient of m*f + p r*g has absolute value $\leq 2d_g(1+p) = 8d_g < q/2$

N	d_g	q
509	127	2048
677	127	2048
821	255	4096

NTRU-1998

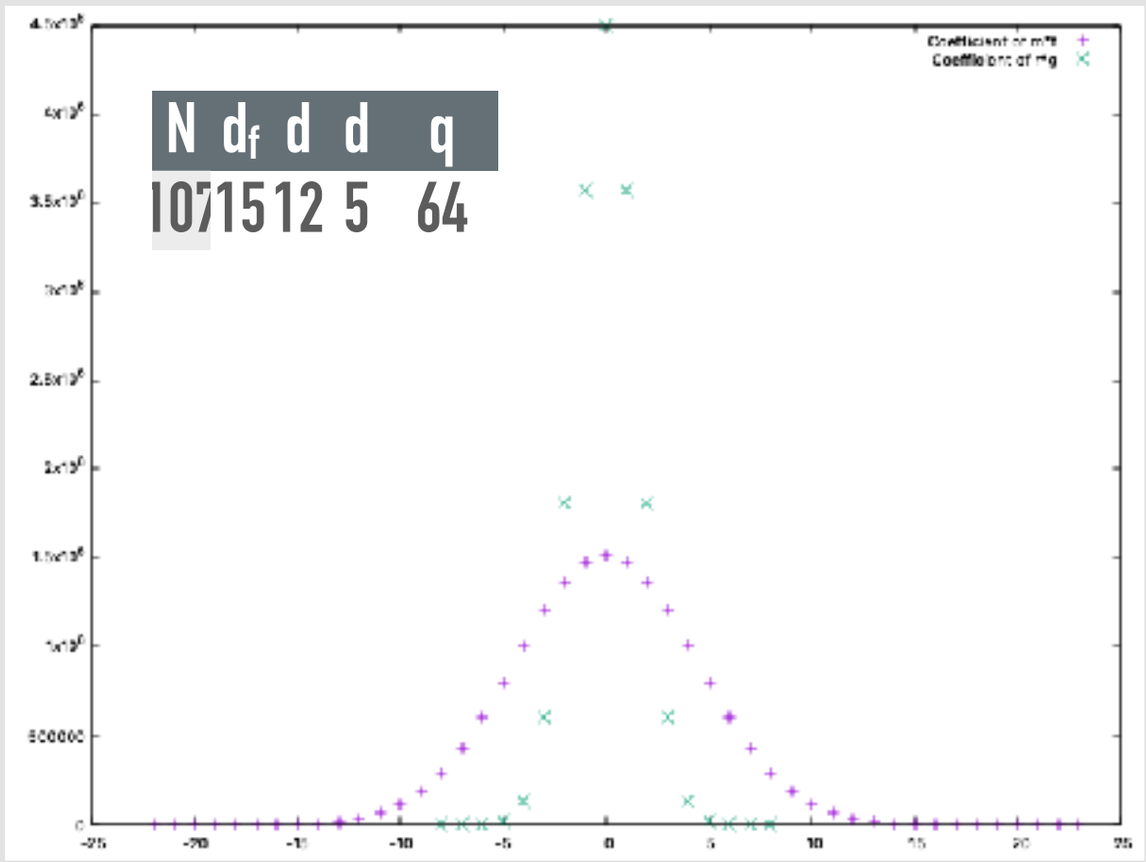
- f has d_f coeffs +1 and d_f-1 coeffs -1, so each coefficient of m*f is in $\{-2d_f+1,\ldots,2d_f-1\}$
- r has d_r coeffs +1 and d_r coeffs -1, so each coefficient of r*g is in $\{-2d_r,\ldots,2\ d_r\}$
- So each coefficient of m*f + p r*g has absolute value ≤ $2d_f-1+2pd_r$ which is ≥ q/2.

N	d_f	d_g	d_r	$2d_f-1+2pd_r$	q/2
107	15	12	5	59	32
167	61	20	18	229	64
503	216	72	55	761	128

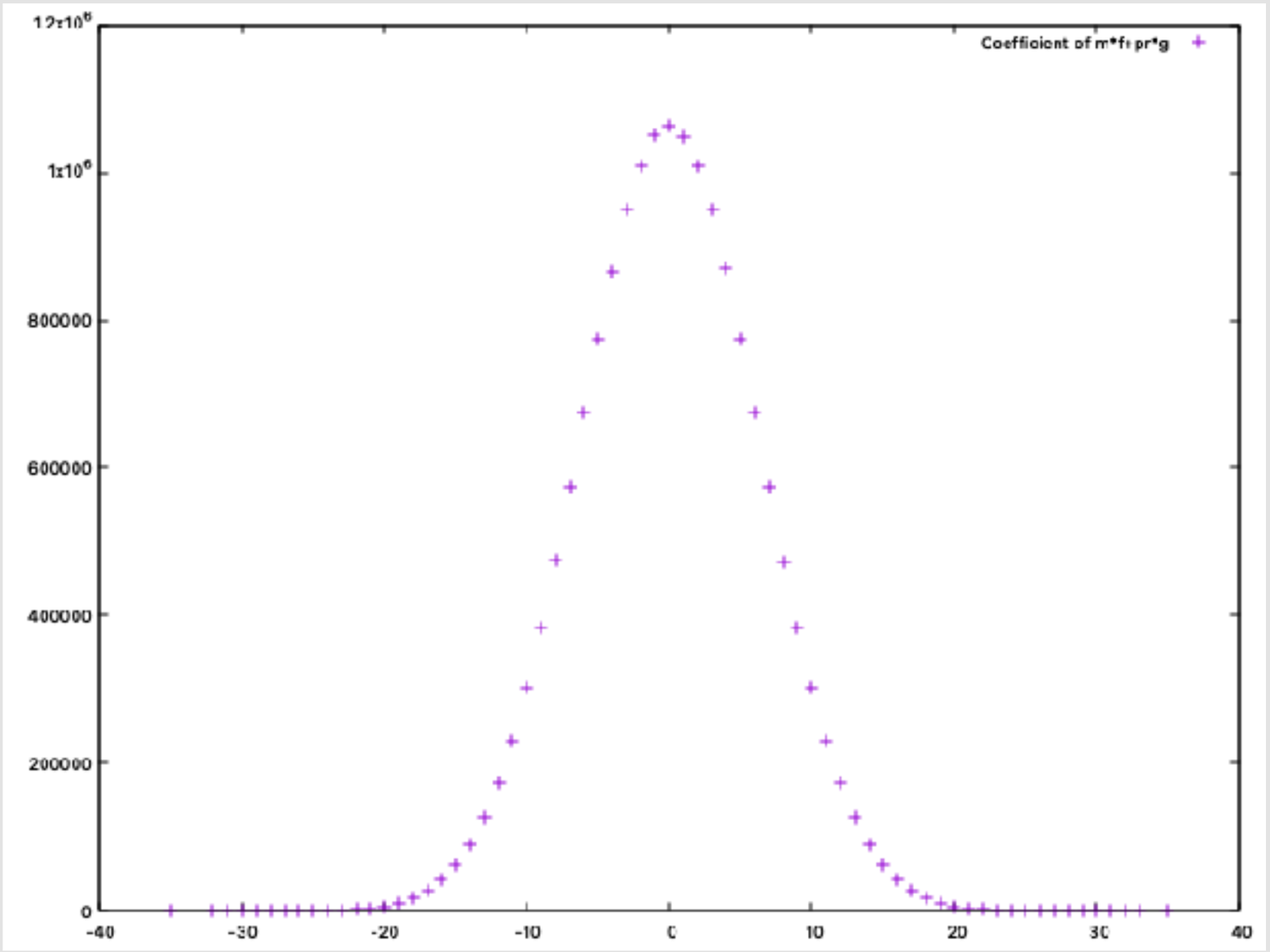
NTRU-1998 DISTRIBUTION

- f has d_f coeffs +1 and d_f-1 coeffs -1, so each coefficient of m*f is a sum of $2d_f$-1 independent uniform variables in $\{0, \pm 1\}$, but the coeffs are not independent.
- r has d_r coeffs +1 and d_r coeffs -1, so each coefficient of r*g is in $\{-2d_r, \ldots, 2\ d_r\}$

N	d_f	d_g	d_r
107	15	12	5
167	61	20	18
503	216	72	55



NTRU-1998 DISTRIBUTION OF COEFFICIENTS



8 million samples

N	q
107	64

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 mod q can be interpreted in terms of lattice.

- The set L of all polynomials u and v in R such that h*u = v mod 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 mod q) in L, because the difference is (pr,m).
- This is a BDD problem like in GGH encryption.

DIFFIE-HELLMAN AND EL GAMAL

Encryption without Trapdoors

TRAPDOOR-LESS ENCRYPTION

DIFFIE-HELLMAN KEY EXCHANGE

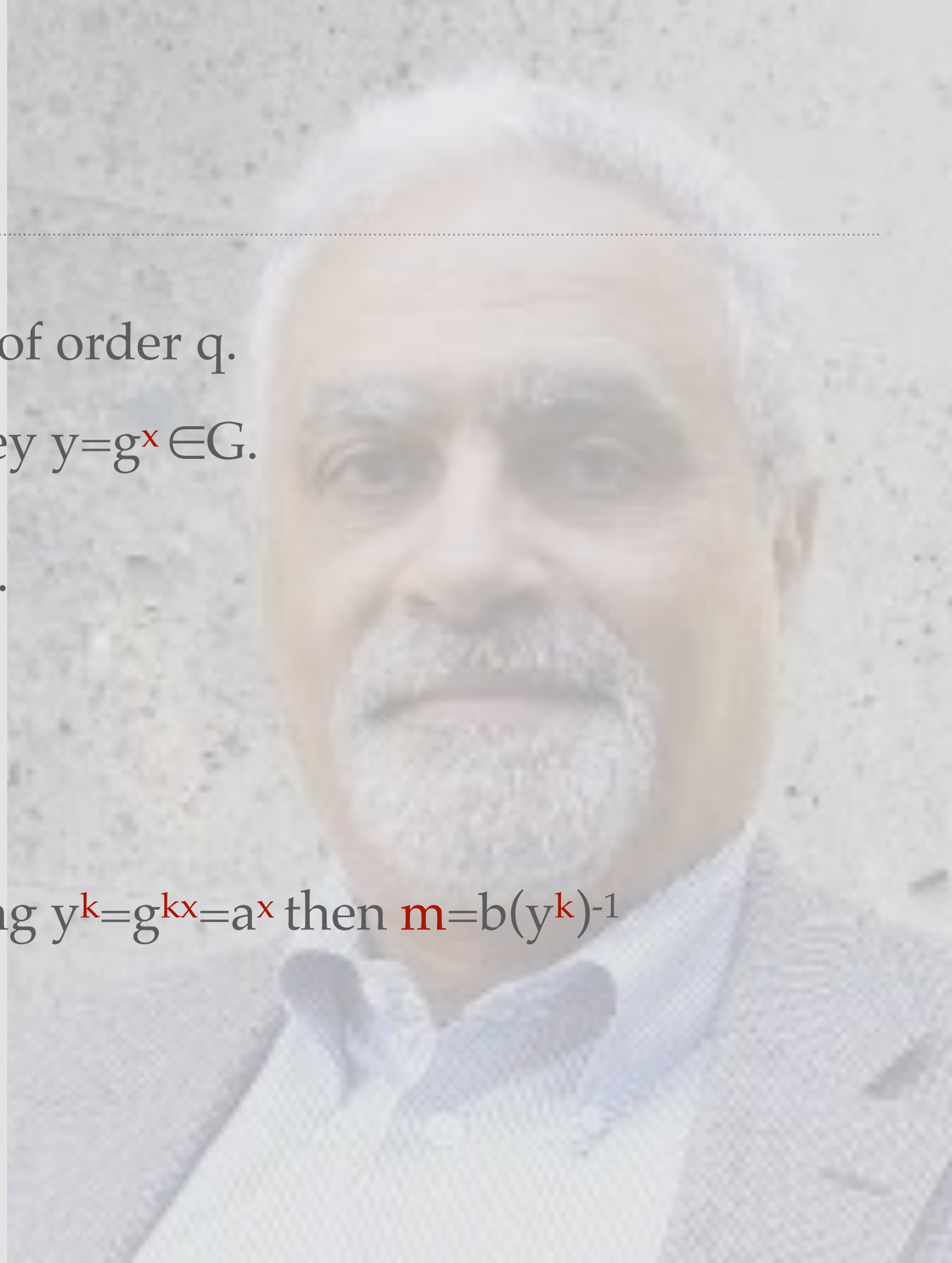
- Let $G=\langle g \rangle$ be generated by g of order q.

- Both can compute the shared key $g^{ab}=(g^a)^b=(g^b)^a$

➤ This key exchange is the core of El Gamal public-key encryption.

ELGAMAL ENCRYPTION (1984)

- Let G be a cyclic group $\langle g \rangle$ of order q.
- Secret key $x \in \mathbf{Z}_q$. Public key $y = g^x \in G$.
- Encrypt $m \in G$ as $(a,b) \in G^2$.
  - $a = g^k \in G$ where $k \in \mathbf{Z}_q$
  - $b = my^k \in G$
- Decrypt (a,b) by recovering $y^k = g^{kx} = a^x$ then $m = b(y^k)^{-1}$



ELGAMAL ENCRYPTION

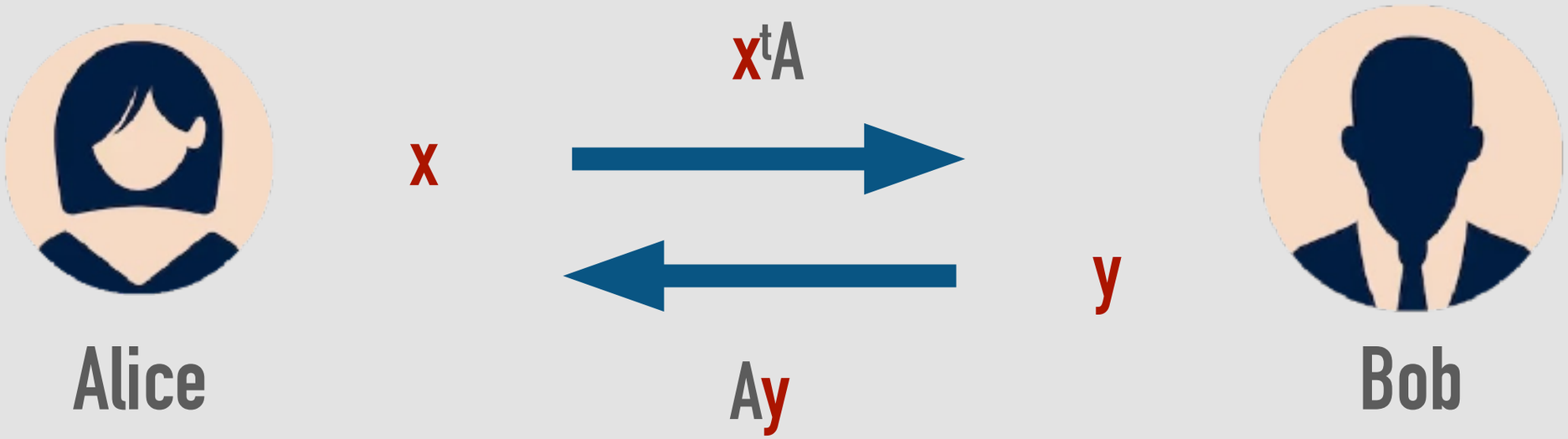
- Behind El Gamal, there is the Diffie-Hellman key exchange.

➤ 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}.

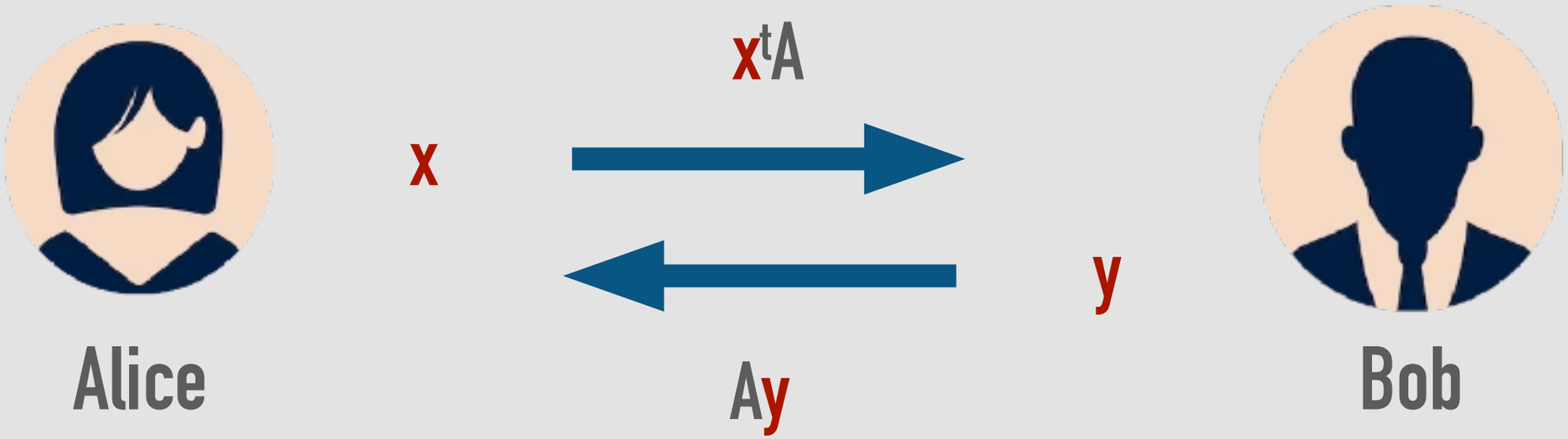
MATRIX DIFFIE-HELLMAN



- A is a public matrix over some ring R. x and y are secret vectors.
- Alice and Bob can compute $x^tAy = (x^tA)y = x^t(Ay) \in R$

➤ Is this secure?

MATRIX DIFFIE-HELLMAN



- Both can compute $x^tAy = (x^tA)y = x^t(Ay) \in R$
- But so would <u>anyone</u> who can compute x' or y' s.t. $x^tA = x'^tA$ or $Ay = Ay'$ which is easy: linear algebra!

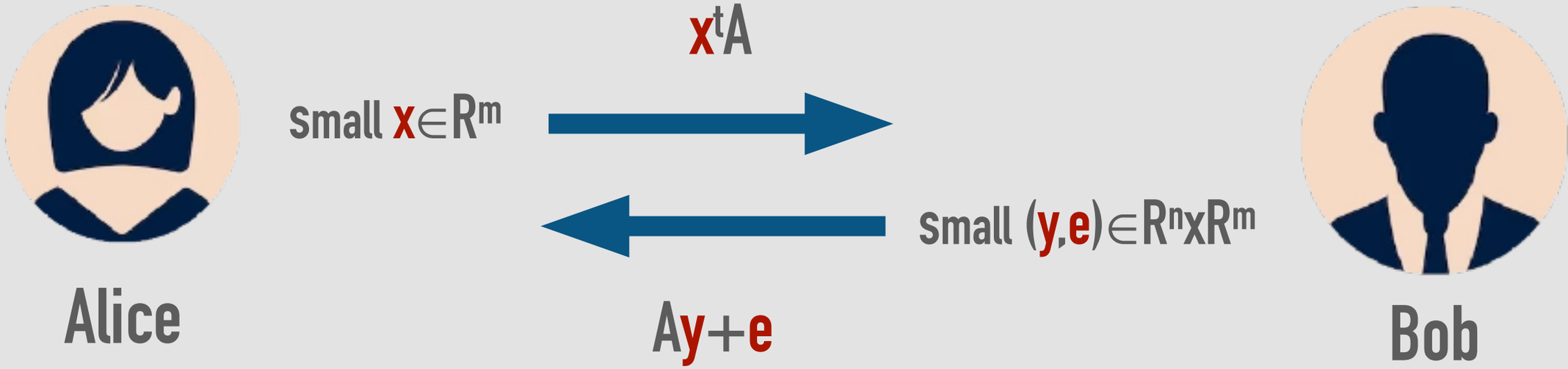
WHAT CAN WE DO?

KEY IDEAS

Introducing…

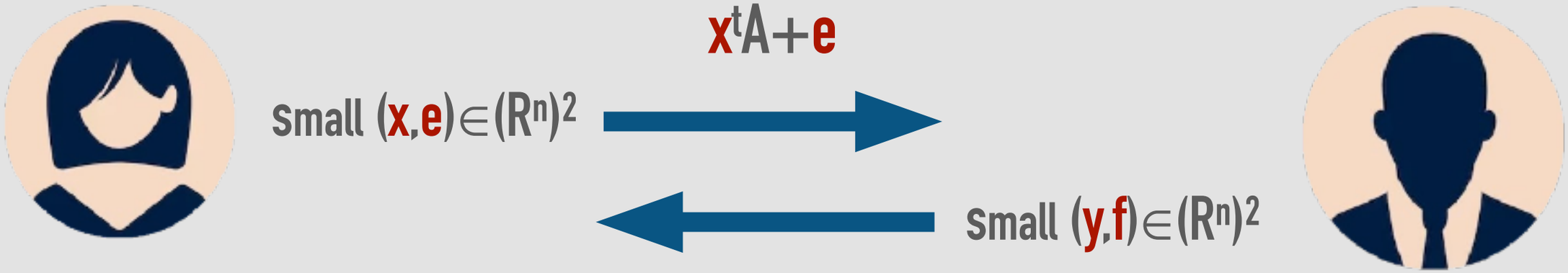
- Noise
- Small secrets

RECTANGULAR DIFFIE-HELLMAN



- Both can approximate $x^tAy \approx x^t(Ay+e)$ if in the ring R:
  - small+small = small
  - small*small = small
- A is an mxn matrix over $R=Z_q$ $m \gg n$

SQUARE DIFFIE-HELLMAN



- Both can approximate $x^tAy \approx x^t(Ay+f) \approx (x^tA+e)y$.
- A is an nxn matrix over $R=\mathbf{Z}_q[X]/(X^{256}+1)$
- This is the noisy Diffie-Hellman underlying Kyber:

  Alice and Bob use MLWE.

WHICH RINGS?

- $R=\mathbf{Z}_q$ or $\mathbf{Z}_q[X]/(X^{256}+1)$ so that:
  - small+small = small
  - small*small = small

- For suitable prime q, $X^{256}+1$ splits into many factors mod q, allowing fast multiplication in $\mathbf{Z}_q[X]/(X^{256}+1)$.

RECALL THE LATTICE MAGIC

THE SIS PROBLEM (1996): SMALL INTEGER SOLUTIONS

- Choose an mxn random matrix A over $R=\mathbf{Z}_q$ $m \gg n$
- Goal: Find « short » $x \in R^m$ s.t. $x^t A \equiv 0$

➤ This is essentially finding a short vector in a random lattice.

THE LWE PROBLEM: LEARNING (A CHARACTER) WITH ERRORS

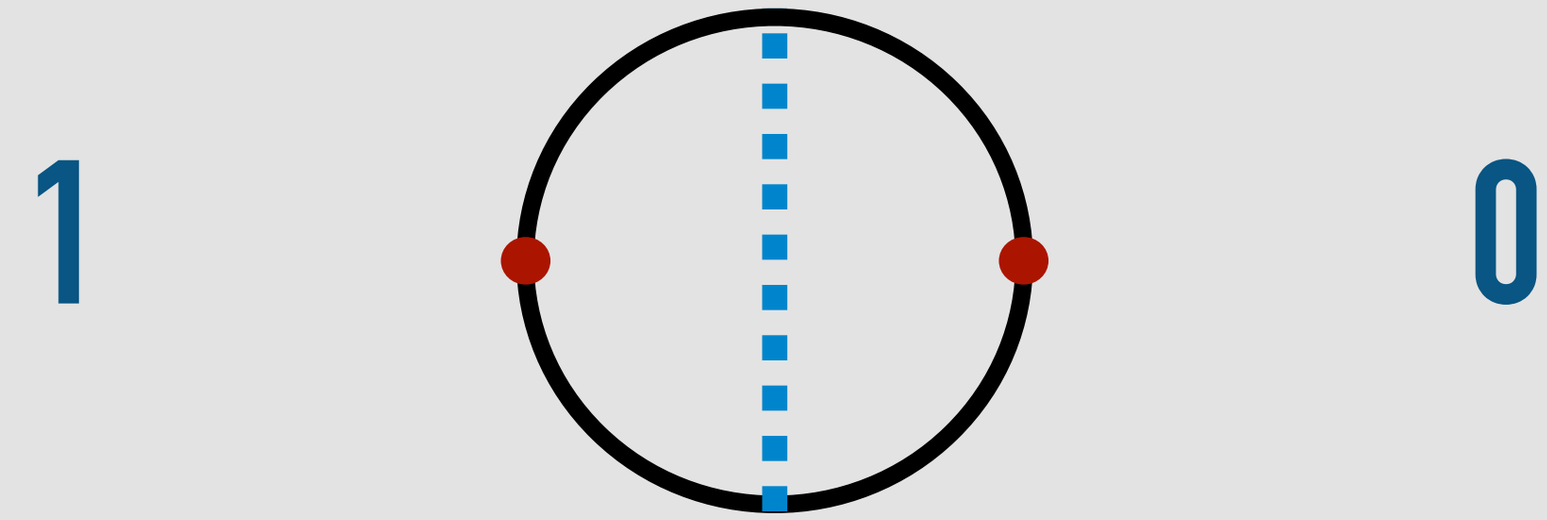
- Choose an mxn random matrix A over $R=\mathbf{Z}_q$ $m \gg n$
- Pick a random y in R^n.

➤ Goal: recover y given Ay+e where $e \in R^m$ is a small noise.

BACK TO NOISY DIFFIE HELLMAN

≠ DIFFIE-HELLMAN: THE NOISE

- The two values computed by Alice and Bob are elements of a torus which are close to each other.

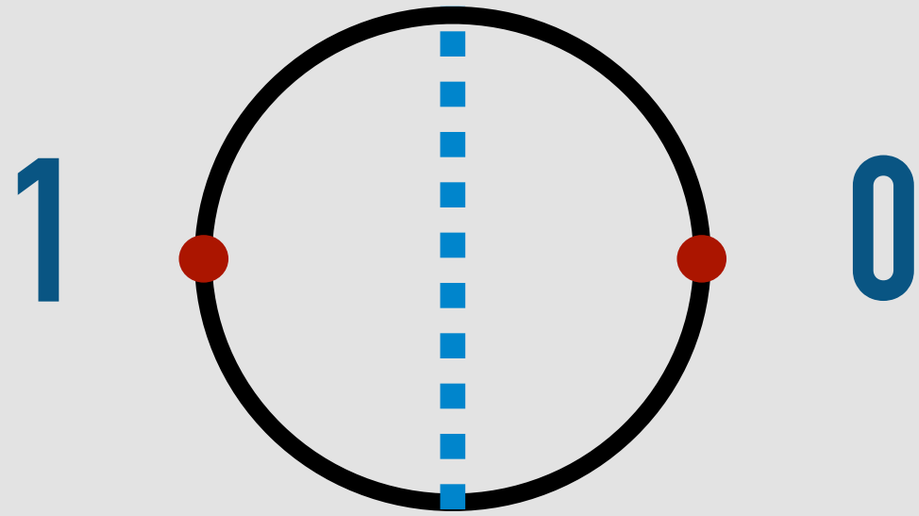


➤ But how can they extract a bit?

Key reconciliation.

NOISY EL GAMAL

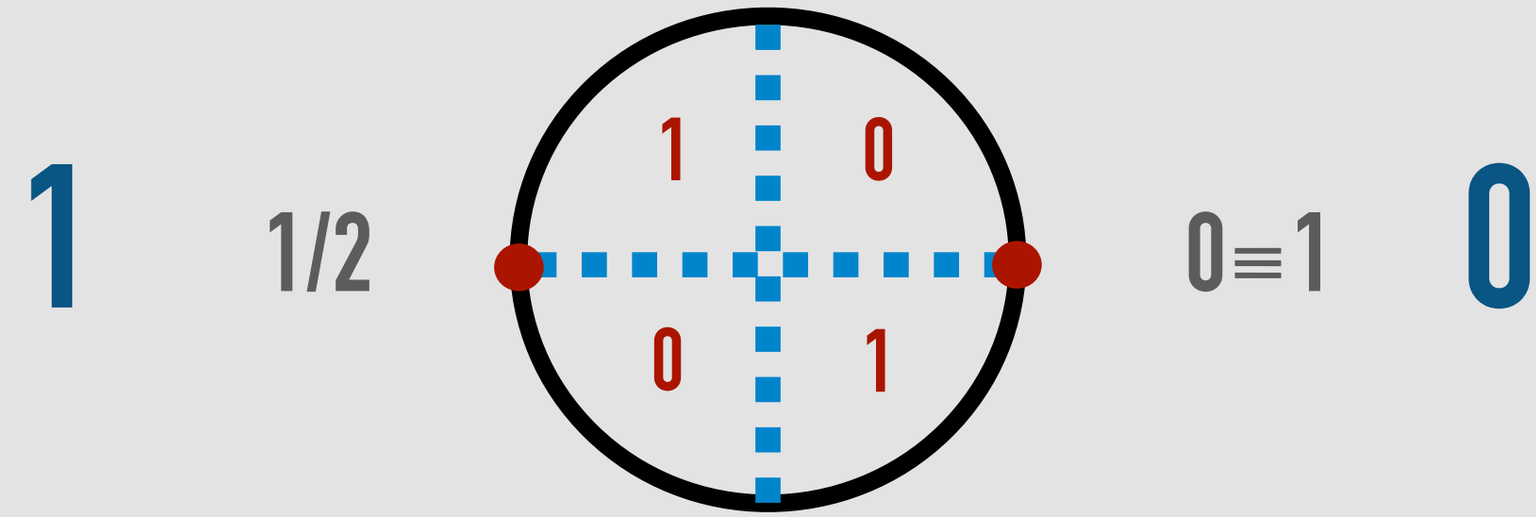
- No problem in El Gamal by encoding the message into torus elements which are far apart.



- $R=\mathbf{Z}_q[X]/(X^{256}+1)$ encodes 256 bits.

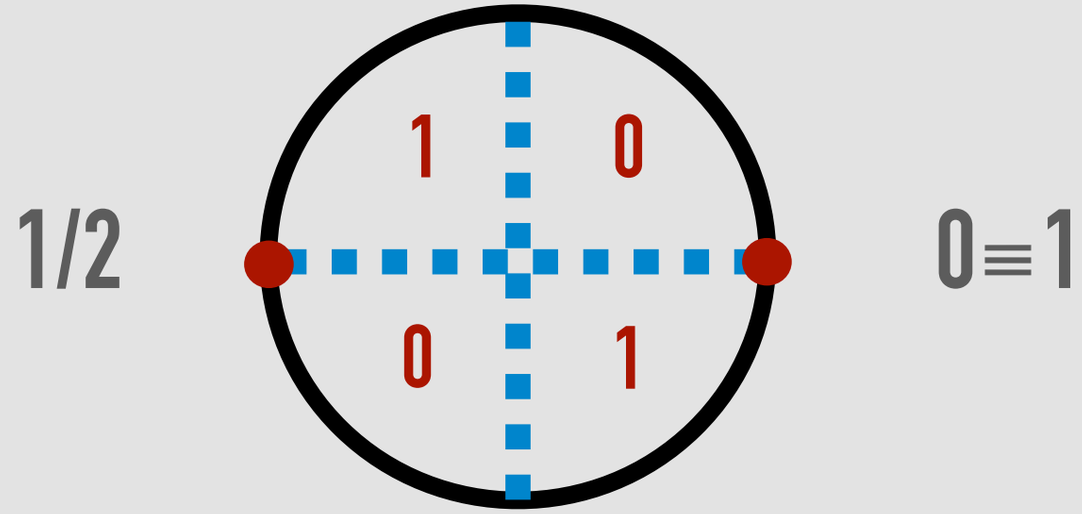
KEY RECONCILIATION

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



KEY RECONCILIATION

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



KYBER SETTINGS

- Ring $\mathbf{Z}_{3329}[X]/(X^{256}+1)$ for fast NTT multiplication: $3329 \equiv 1 \pmod{256}$
- 2x2, 3x3 and 4x4 matrices
- Decryption failure probability $\leq 2^{-139}$
- Small distribution = binomial over {-2,…,2} = 0/1 + 0/1 - 0/1 - 0/1

ANOTHER LOOK

ABSTRACTING DH

- Let e: (a,b) $\mapsto g^{ab}$.

  This map is a pairing:

  $\mathbf{Z}_q \times \mathbf{Z}_q \to G$ is bilinear.

- Let f: a $\mapsto g^a$ be the DL one-way function

  $\mathbf{Z}_q \to G$

- 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),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 $\mathbf{Z}^m$.
- $f_g(x_1,\ldots,x_m)=\Sigma_i\, x_i\, g_i \in G$.

➤ Inversion is as hard as SIS.

THE LWE ONE-WAY FUNCTION

- Let $g_1,\ldots,g_m$ 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 $f^x_g(s,e)= (s(g_1),\ldots,s(g_m))+e \in(\mathbf{R}/\mathbf{Z})^m)^m$

➤ Inversion is LWE.

PAIRING FROM LATTICES

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

  The dual group G^x induces a pairing $G^x \times \mathbf{Z}^m \to \mathbf{R}/\mathbf{Z}$

  by $\varepsilon(s,(x_1,\ldots,x_m)) = s(\Sigma_i\, x_i\, g_i)$

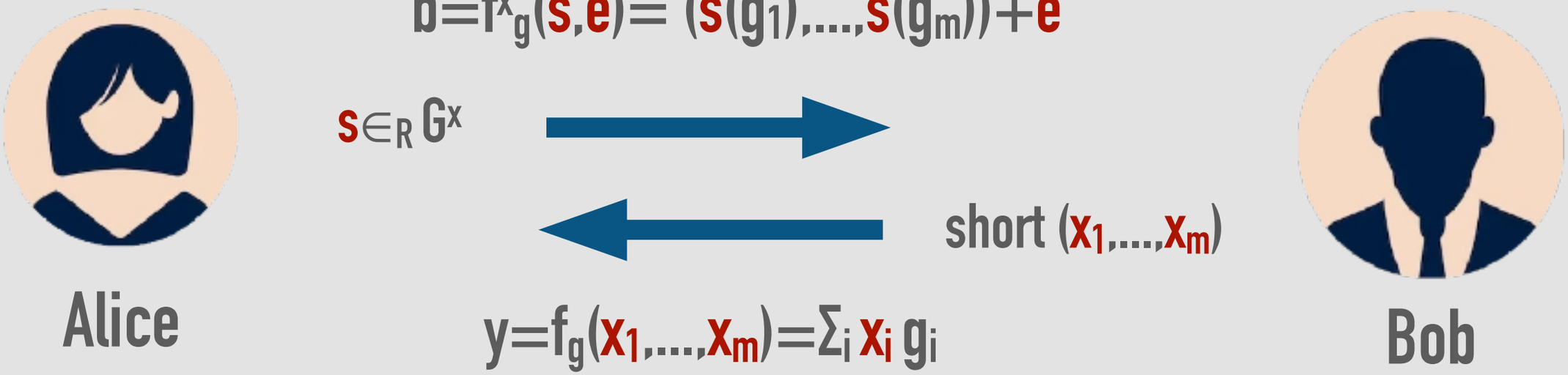
- Let $y=f_g(x_1,\ldots,x_m)=\Sigma_i\, x_i\, g_i \in G$ where x_i's small,

  and $b=f^x_g(s,e)= (s(g_1),\ldots,s(g_m))+e \in(\mathbf{R}/\mathbf{Z})^m$, 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.



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

  Alice computes $s(y)+e'$ and Bob computes $\Sigma_i\, x_i\, b_i$.

➤ Security is related to DDH.

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 $s \in_R G^x$

  Public key $b=f^x_g(s,e)= (s(g_1),\ldots,s(g_m))+e$

- Encrypt $m \in \{0,1\}$ as $(y,c) \in Gx(\mathbf{R}/\mathbf{Z})$
- $y = f_g(x_1,\ldots,x_m)=\Sigma_i\, x_i\, g_i$ where $(x_1,\ldots,x_m)$ is short
- $c = s(y) + e' + (m/2)$
- 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 $y=f_g(x_1,\ldots,x_m)=\Sigma_i\, x_i\, g$

- Encrypt $m\in\{0,1\}$ as $(b,c)\in(\mathbf{R}/\mathbf{Z})^m x(\mathbf{R}/\mathbf{Z})$
- $b=f^x_g(s,e)=(s(g_1),\ldots,s(g_m))+e$ where $s\in_R G^x$
- $c=\Sigma_i\, x_i\, b_i+(m/2)$
- 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?