Fully Homomorphic Encryption Using Ideal Lattices

Craig Gentry

Stanford University, IBM

Fields Institute, 05/11/09

Wouldn't it be neat if you could…

Query encrypted data?

- *Store your encrypted data* on an untrusted server
- *Query* the data – i.e., make boolean queries on the data
- *Get a useful response* from the server, without the server just sending all of the data to you

Wouldn't it be neat if you could…

Query encrypted data?

- *Store your encrypted data* on an untrusted server
- *Query* the data – i.e., make boolean queries on the data
- *Get a useful response* from the server, without the server just sending all of the data to you

Query data privately?

- *Send an encrypted query* regarding stored data (e.g., on Google's servers)
- *Get a useful concise response*

Wouldn't it be neat if you could…

Query encrypted data?

- *Store your encrypted data* on an untrusted server
- *Query* the data – i.e., make boolean queries on the data
- *Get a useful response* from the server, without the server just sending all of the data to you

Query data privately?

- *Send an encrypted query* regarding stored data (e.g., on Google's servers)
- *Get a useful concise response*

Do both simultaneously?

Privacy Homomorphism (a.k.a. Fully Homomorphic Encryption)

Well, here's how:

- Privacy homomorphism: Rivest, Adleman and Dertouzos proposed the concept in 1978. (Rivest, Shamir, and Adleman proposed RSA in 1977, published in 1978.)
- Assume you have public-key encryption scheme that, in addition to algorithms (KeyGen, Enc, Dec), has an efficient algorithm "Evaluate", such that:

$$\text{Evaluate}(pk, C, \psi_1, \ldots, \psi_t) \approx \text{Enc}(pk, C(\pi_1, \ldots, \pi_t))$$

for all pk, all circuits C, all $\psi_i = \text{Encrypt}(pk, \pi_i)$.

Privacy Homomorphism

Well, here's how:

- Assume you have public-key encryption scheme that, in addition to algorithms (KeyGen, Enc, Dec), has an efficient algorithm "Evaluate", such that:

$$\text{Evaluate}(pk, C, \psi_1, \ldots, \psi_t) \approx \text{Enc}(pk, C(\pi_1, \ldots, \pi_t))$$

for all pk, all circuits C, all ψ_i = Encrypt(pk, π_i).

Query encrypted data:

Encrypt stored data: $\psi_1, \ldots, \psi_t$

Query: send your circuit C

Response: Eval(pk, C, $\psi_1, \ldots, \psi_t$)

Decrypt response $\rightarrow C(\pi_1, \ldots, \pi_t)$

Privacy Homomorphism

Well, here's how:

- Assume you have public-key encryption scheme that, in addition to algorithms (KeyGen, Enc, Dec), has an efficient algorithm "Evaluate", such that:

$$\text{Evaluate}(pk, C, \psi_1, \ldots, \psi_t) \approx \text{Enc}(pk, C(\pi_1, \ldots, \pi_t))$$

for all pk, all circuits C, all ψ_i = Encrypt(pk, π_i).

Query encrypted data:

Encrypt stored data: $\psi_1, \ldots, \psi_t$

Query: send your circuit C

Response: Eval(pk, C, $\psi_1, \ldots, \psi_t$)

Decrypt response $\rightarrow C(\pi_1, \ldots, \pi_t)$

Query data privately:

Send enc. queries ψ_i = Enc(pk, π_i)

Server uses search circuit C_{data}

Response: Eval(pk, C_{data}, $\psi_1, \ldots, \psi_t$)

Decrypt response $\rightarrow C_{data}(\pi_1, \ldots, \pi_t)$

The Quest for Privacy Homomorphisms

Problem is: We have no such encryption scheme.

- What we have currently:
  - Multiplicatively homomorphic schemes: RSA, ElGamal, etc.
  - Additively homomorphic schemes: GM, Paillier, etc.
  - Quadratic formulas: BGN
  - NC1: SYY
- What we don't have:
  - A *fully homomorphic* scheme for arbitrary circuits

Fully Homomorphic Encryption: Construction

3 Steps

- Step 1 – Bootstrapping:

$$\begin{tikzcd}\boxed{\text{Scheme E can evaluate its own decryption circuit}} \arrow[r] & \boxed{\text{Scheme E* can evaluate any circuit}}\end{tikzcd}$$

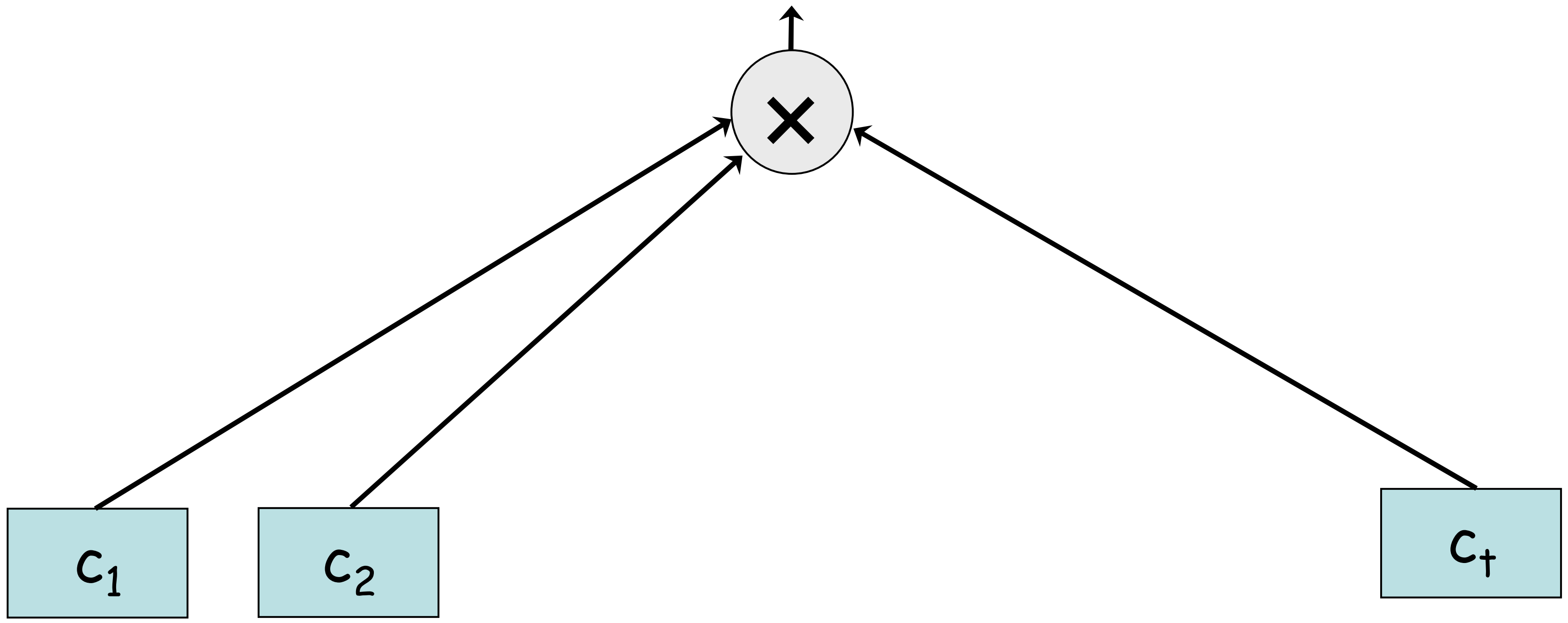
- Step 2 – Ideal Lattices: Decryption in lattice-based systems has low circuit complexity. *Ideal* lattices used to get + and × ops.

- Step 3 – Squashing the Decryption Circuit: the encrypter helps make decryption circuit smaller by starting decryption itself! Like server-aided decryption.

Step 1: Bootstrapping

What Circuits can RSA "Evaluate"?

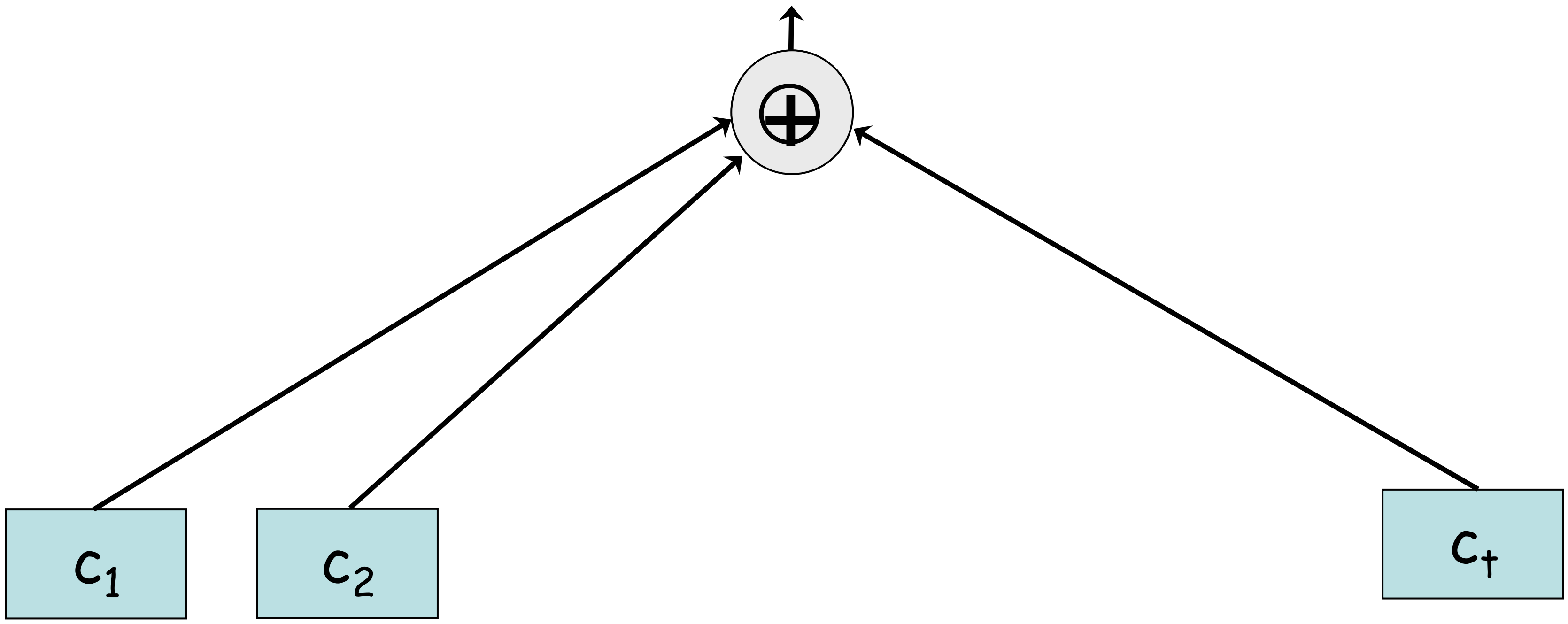
$c \leftarrow c_1 \times c_2 \bmod N, \qquad c = (m_1 \times m_2)^e \bmod N$



A circuit of multiplication (mod N) gates

What Circuits can Goldwasser-Micali "Evaluate"?

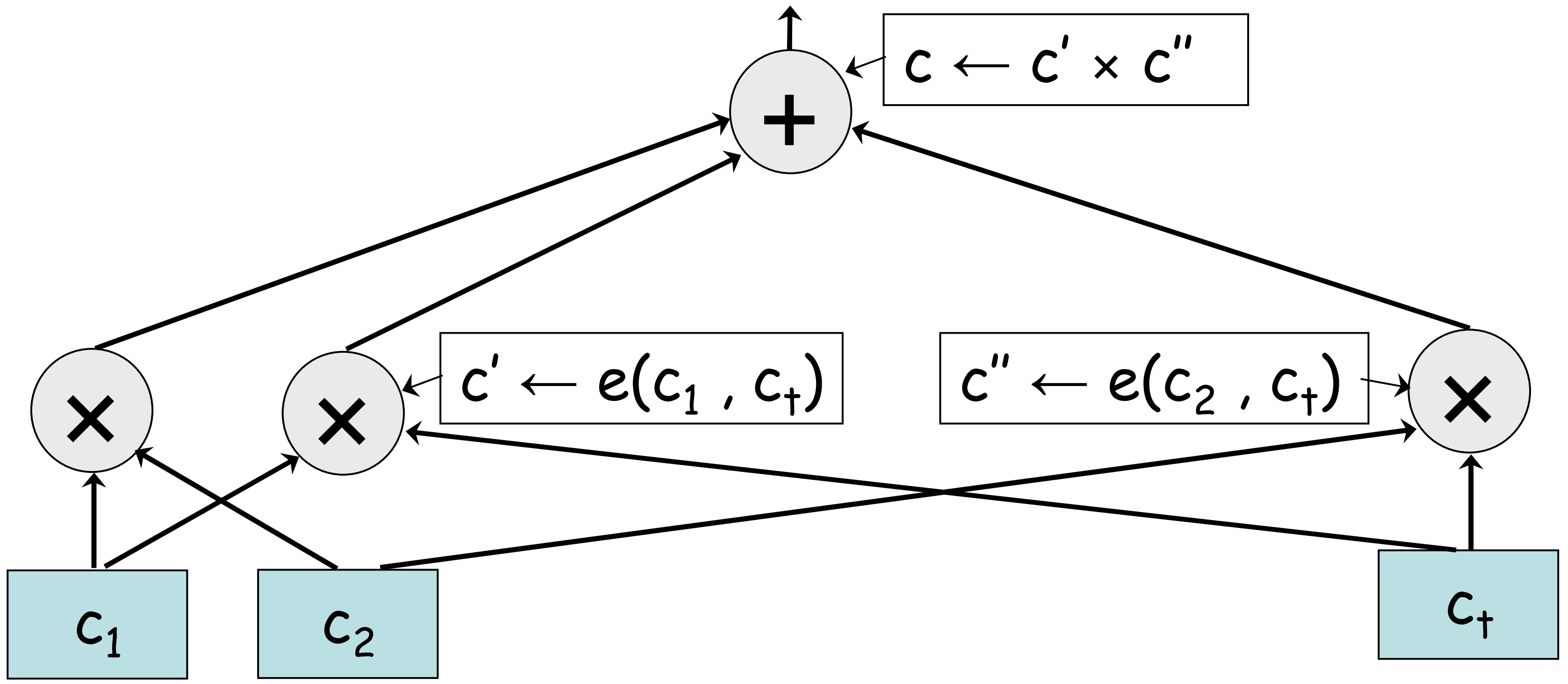
$c \leftarrow c_1 \times c_2 \bmod N, \qquad c = r^2 \times x^{m_1+m_2} \bmod N$



A circuit of XOR gates

What Circuits can Boneh-Goh-Nissim "Evaluate"?

Uses a bilinear map or "pairing": $e : G \times G \rightarrow G_T$


$c \leftarrow c' \times c''$
$c' \leftarrow e(c_1, c_t)$
$c'' \leftarrow e(c_2, c_t)$
$+$
$\times$
$\times$
$\times$
c_1
c_2
c_t

A quadratic formula

Fully Homomorphic Encryption: Informal Definition

Can "evaluate" any circuit

- A too-strong definition (indistinguishable distributions):

$$\mathrm{Evaluate}(pk, C, \psi_1, \ldots, \psi_t) \approx \mathrm{Enc}(pk, C(\pi_1, \ldots, \pi_t))$$

  for all circuits C, all (sk,pk), and $\psi_i = \mathrm{Encrypt}(pk, \pi_i)$.
- Indistinguishability unnecessary for many apps.
- But we can achieve this…

Fully Homomorphic Encryption: Informal Definition

Can "evaluate" any circuit

- What we want:
  - Correctness:

$$\mathrm{Dec}(sk, \mathrm{Evaluate}(pk, C, \psi_1, \ldots, \psi_t)) \quad = \quad C(\pi_1, \ldots, \pi_t)$$

for all circuits C, all (sk,pk), and $\psi_i = \mathrm{Encrypt}(pk, \pi_i)$.

Fully Homomorphic Encryption: Informal Definition

Can "evaluate" any circuit

- What we want:
  - Correctness:

    $\mathrm{Dec}(sk, \mathrm{Evaluate}(pk, C, \psi_1, \ldots, \psi_t)) \;=\; C(\pi_1, \ldots, \pi_t)$

    for all circuits C, all (sk,pk), and $\psi_i = \mathrm{Encrypt}(pk, \pi_i)$.
  - Compactness:
    - Output of Evaluate is *short*.
    - The trivial solution doesn't count:

      $\mathrm{Evaluate}(pk, C, \psi_1, \ldots, \psi_t) \;\rightarrow\; (C, \psi_1, \ldots, \psi_t)$
    - Our requirement: Size of decryption circuit is a *fixed polynomial in security parameter*

A “Complete” Set of Circuits?

A Steppingstone?

- Given: a scheme E that Evaluates some set S of circuits
- Is S complete?: From E, can we construct a scheme that works for circuits of arbitrary depth?

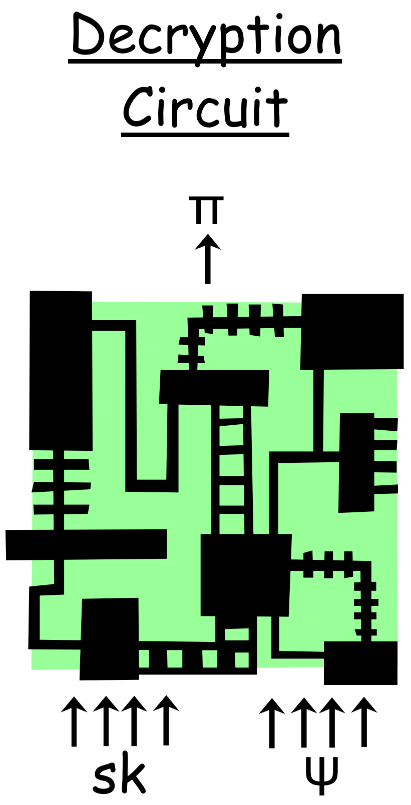
A “Complete” Set of Circuits?

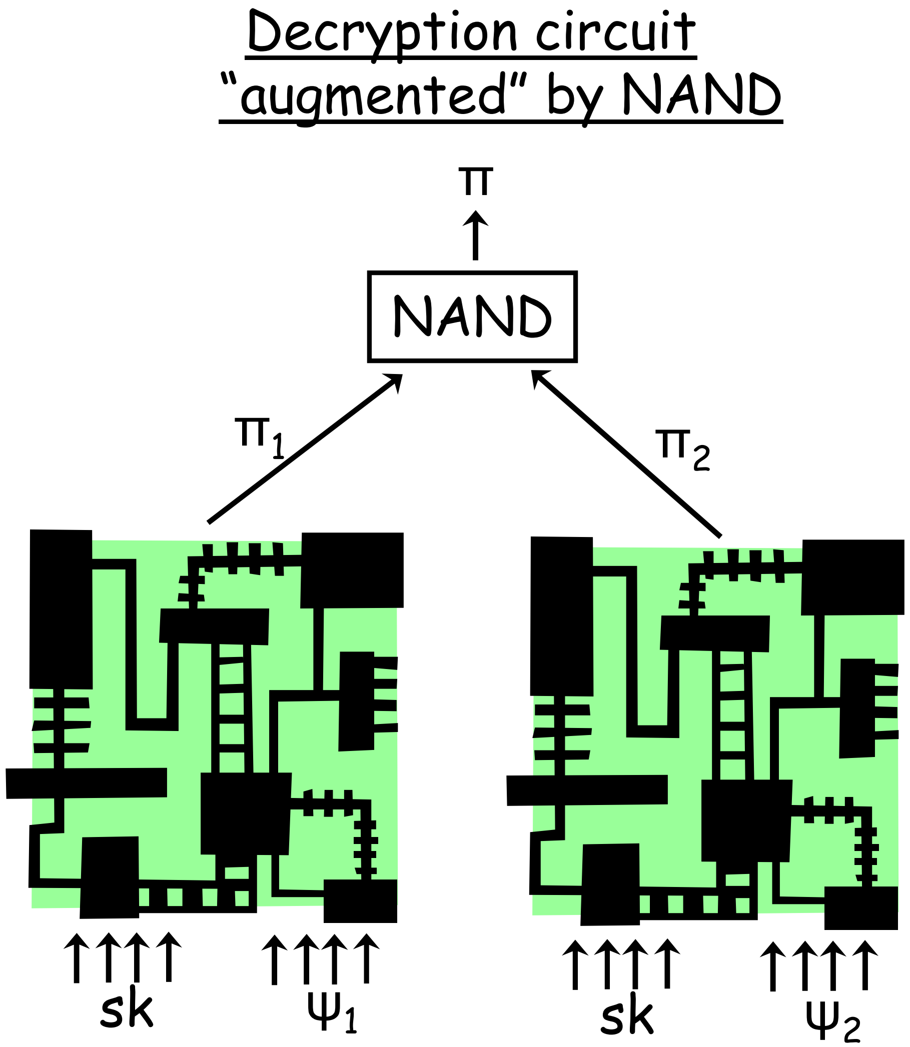
A Steppingstone?

- Given: a scheme E that Evaluates some set S of circuits
- Is S complete?: From E, can we construct a scheme that works for circuits of arbitrary depth?

Yes!

A "Complete" Set of Circuits



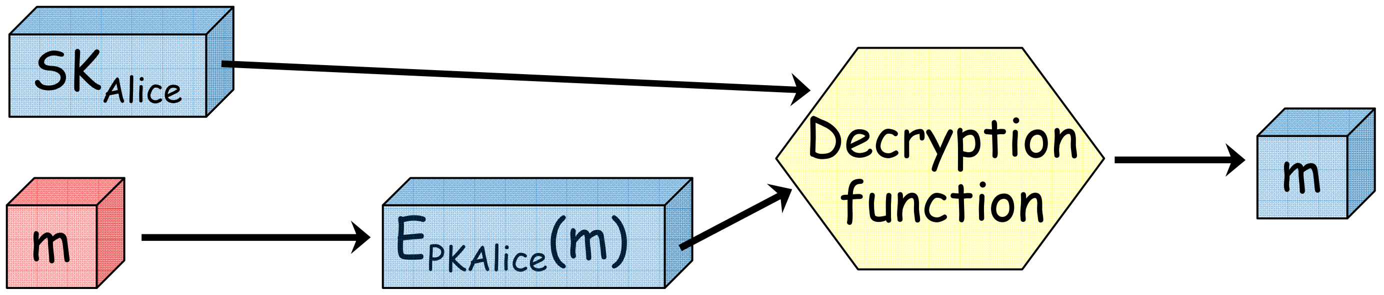


Why is homomorphically evaluating the decryption circuit so powerful?

- Proxy re-encryption: Alice enables anyone to convert a ciphertext under PK_{Alice} to one under PK_{Bob}:

Blue means encrypted under PK_{Bob}.

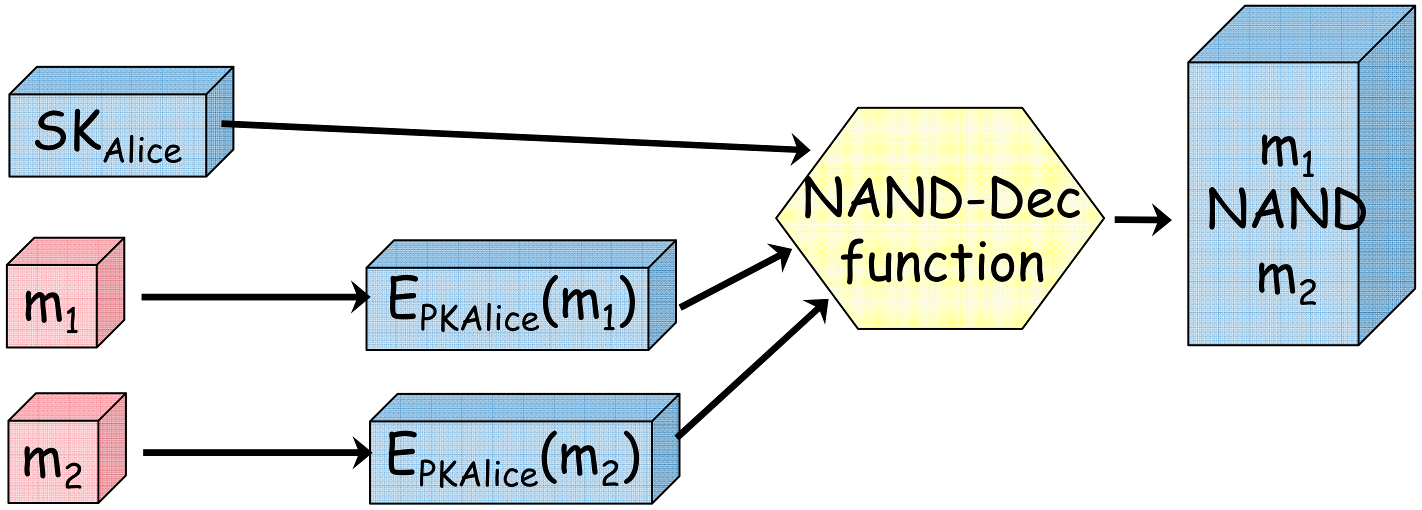
Red means encrypted under PK_{Alice}.



If you can evaluate NAND-Dec…

Blue means encrypted under PK_{Bob}.

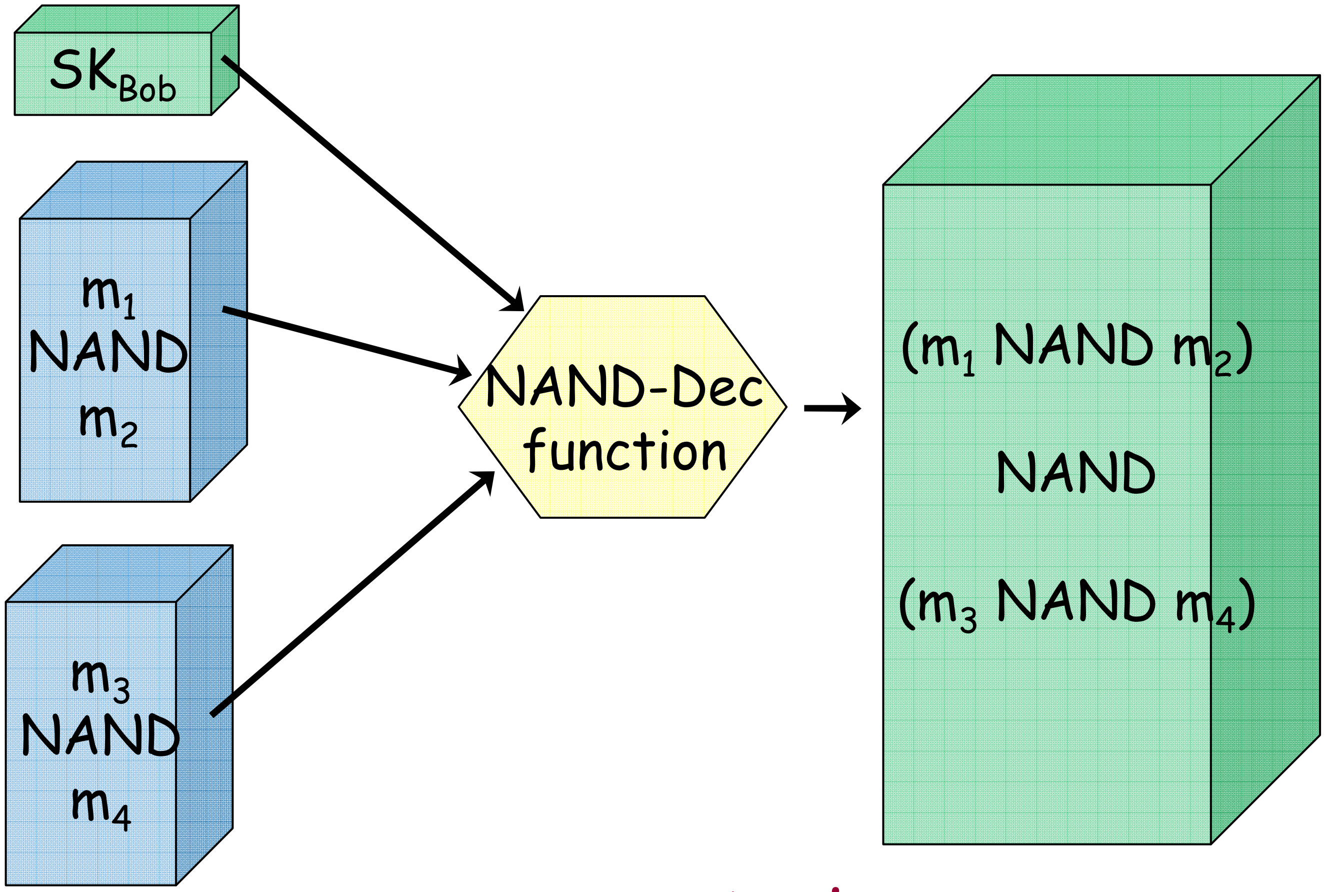
Red means encrypted under PK_{Alice}.



If you can evaluate NAND-Dec

Blue means encrypted under PK_{Bob}.

Green means encrypted under PK_{Carol}.



And so on…

Circuits of Arbitrary Depth

Theorem (informal):

- Suppose scheme E is bootstrappable – i.e., it evaluates its own decryption circuit augmented by gates in Γ.
- Then, there is a scheme E_δ that evaluates arbitrary circuits of depth δ with gates in Γ.
- Ciphertexts: Same size in E_δ as in E.
- Public key:
  - Consists of (δ+1) E pub keys: $pk_0, \ldots, pk_\delta$
  - Along with δ encrypted secret keys: $\{Enc(pk_i, sk_{(i-1)})\}$
  - Linear in δ.
  - Constant in δ, if you assume encryption is "circular secure."

Step 2: Ideal Lattices

Our Task Now…

Find an encryption scheme E that can evaluate its own decryption circuit, plus some.

Our Task Now…

Find an encryption scheme E that can evaluate its own decryption circuit, plus some.

Bootstrappability gives us a new angle:

- Don't just *maximize* the scheme's "*evaluative capacity*"
- Also *minimize* the *circuit complexity of decryption*

Our Task Now…

Find an encryption scheme E that can evaluate its own decryption circuit, plus some.

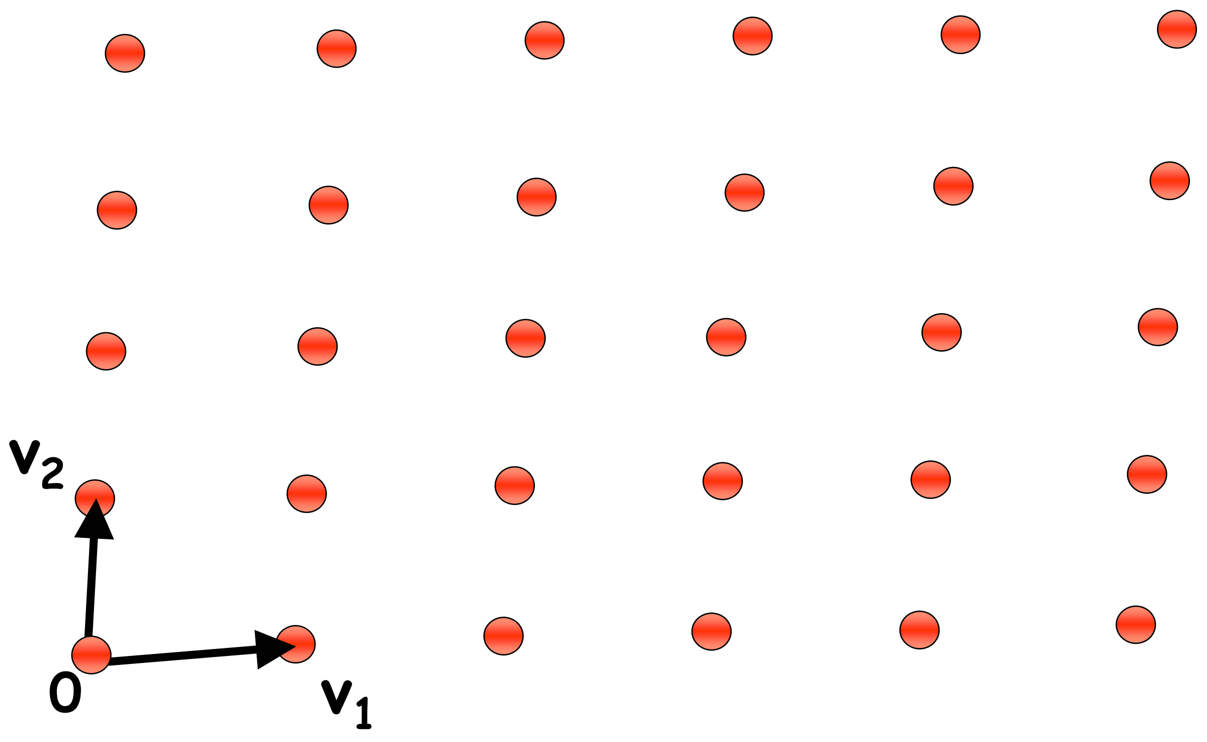
Bootstrappability gives us a new angle:

- Don't just *maximize* the scheme's "*evaluative capacity*"
- Also *minimize* the *circuit complexity of decryption*

Where to Look?:

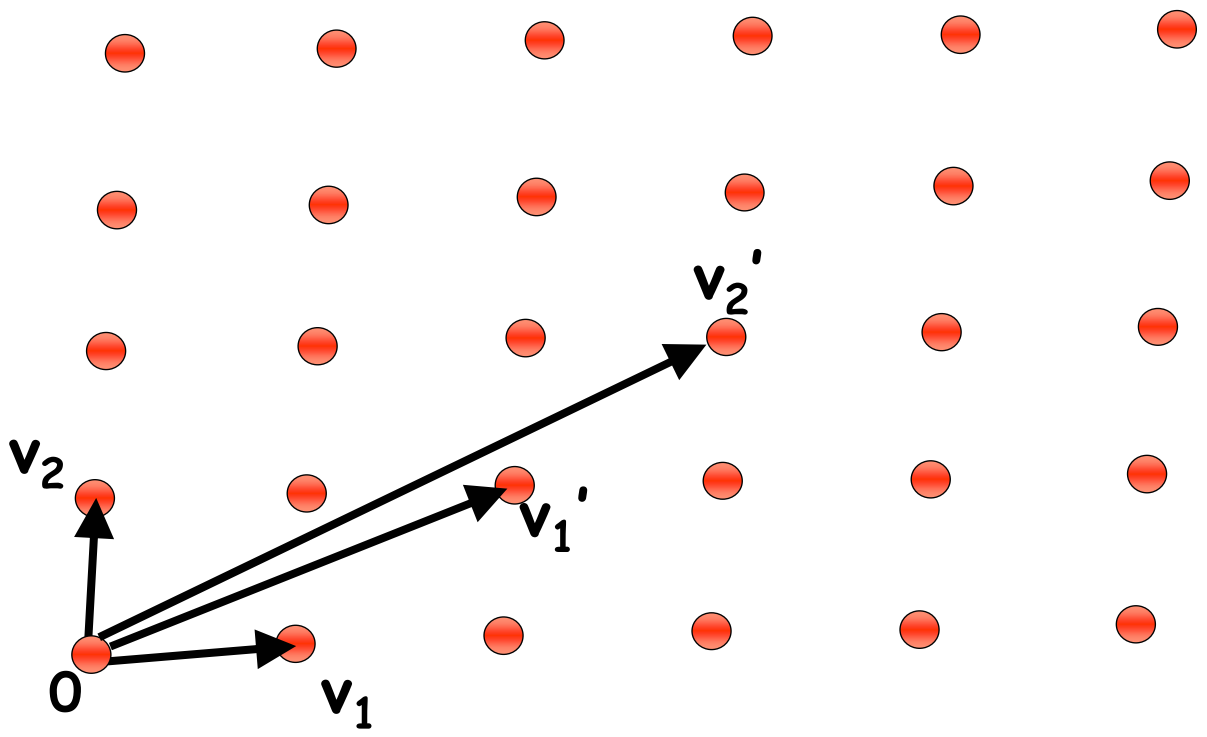
- Not RSA: Exponentiation is highly unparallelizable – i.e., it requires deep circuits
- Maybe schemes based on codes or *lattices*…
  - "Decoding" is typically an inner product – parallelizable!

What's a Lattice?



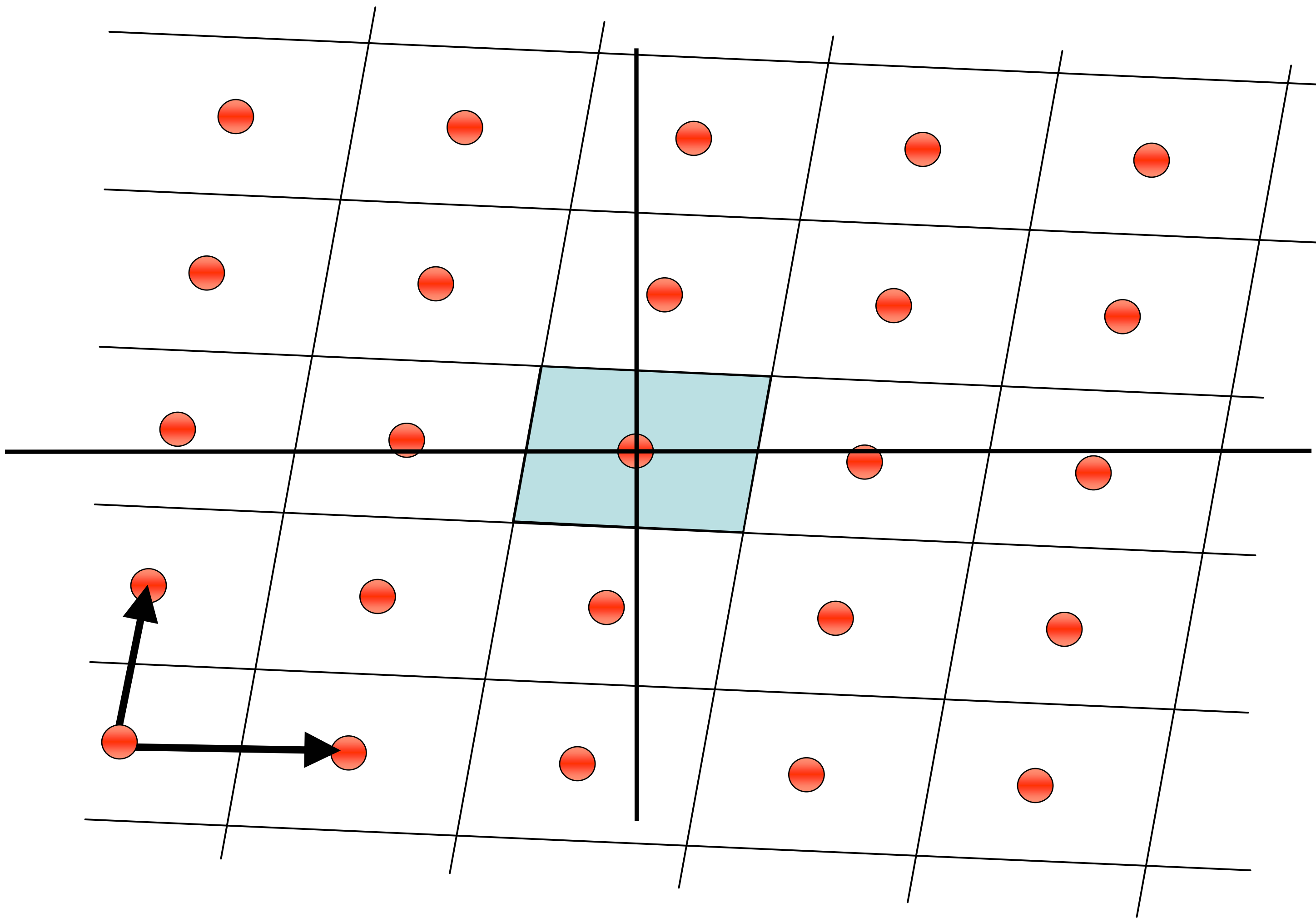
A set of points, or vectors, that looks like this.

What's a Lattice?

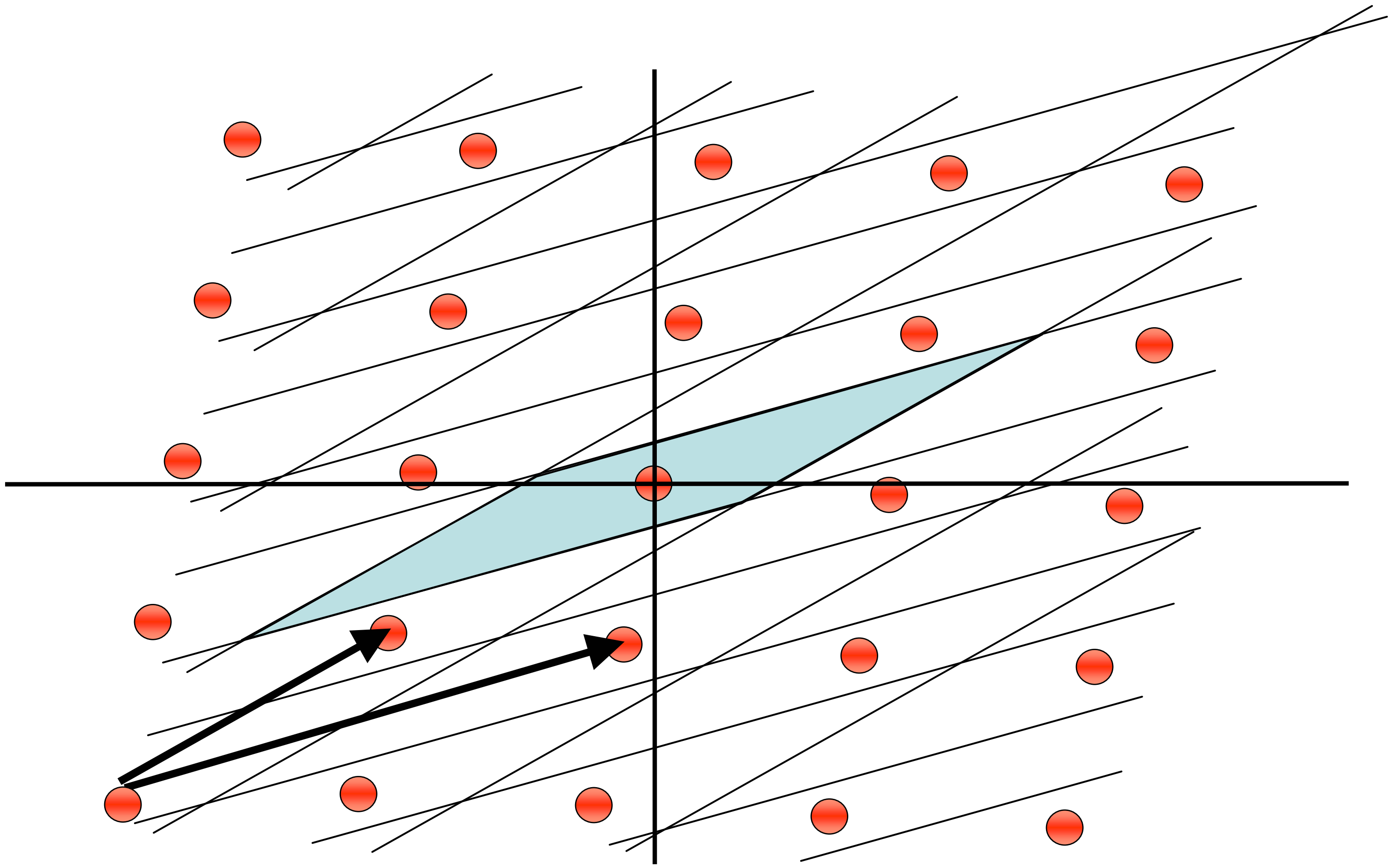


- (v_1, v_2) is a *basis* of the lattice L, since L = { $x_1v_1 + x_2v_2$: x_i in Z (integers) }
- Bases are not unique
- (v_1, v_2) looks like a better basis, don't you think?

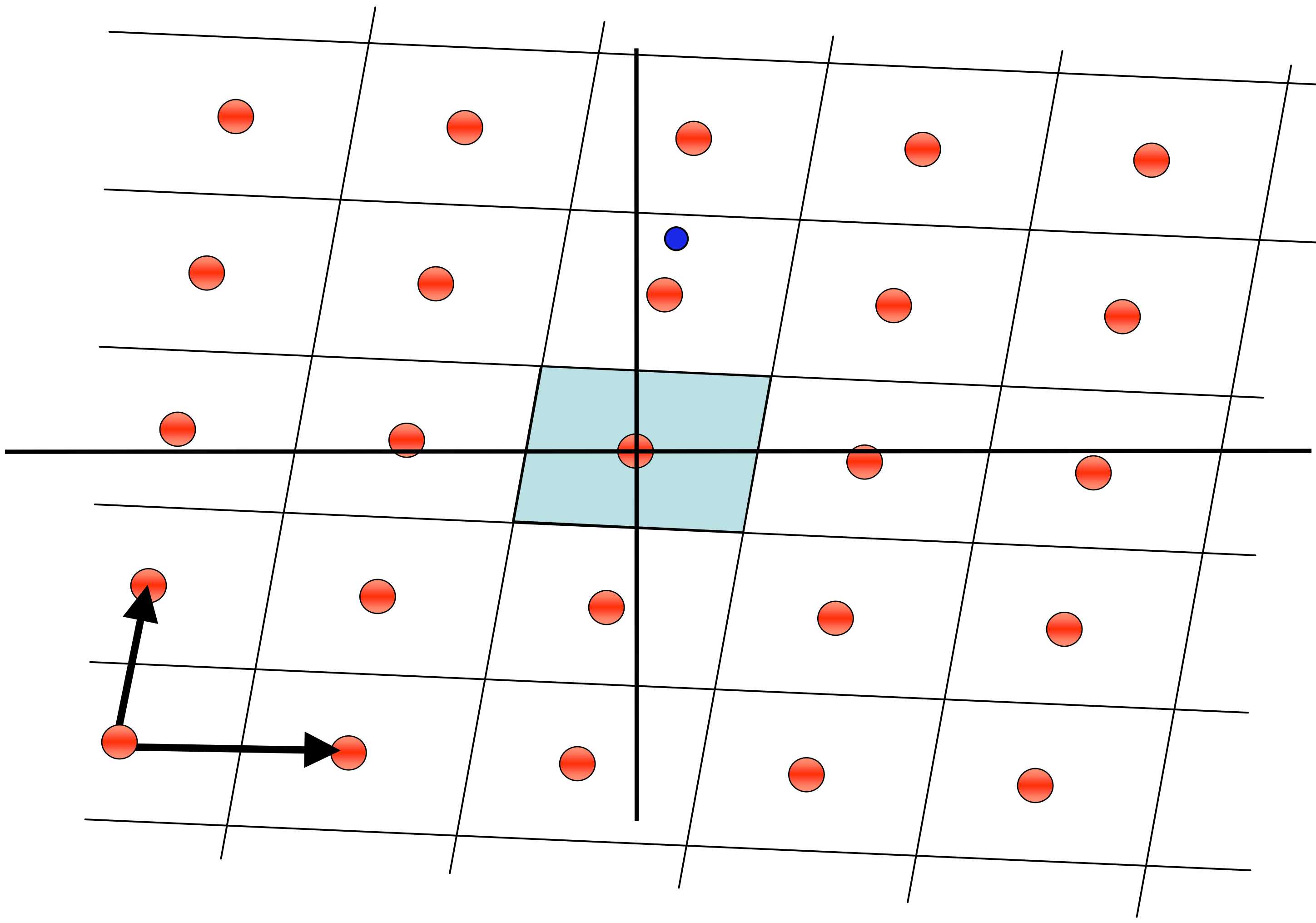
Parallelepipeds



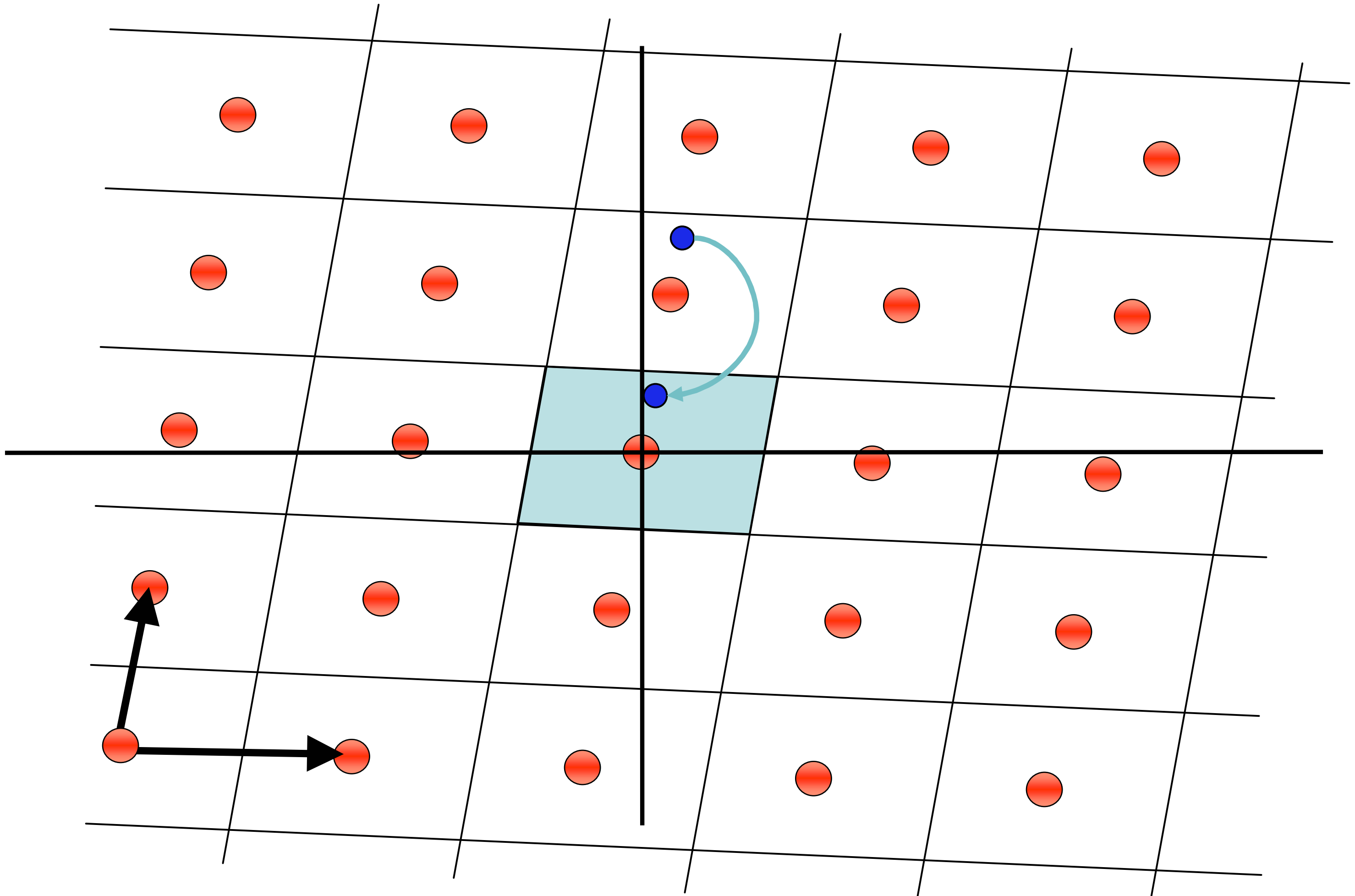
Parallelepipeds



Good Basis

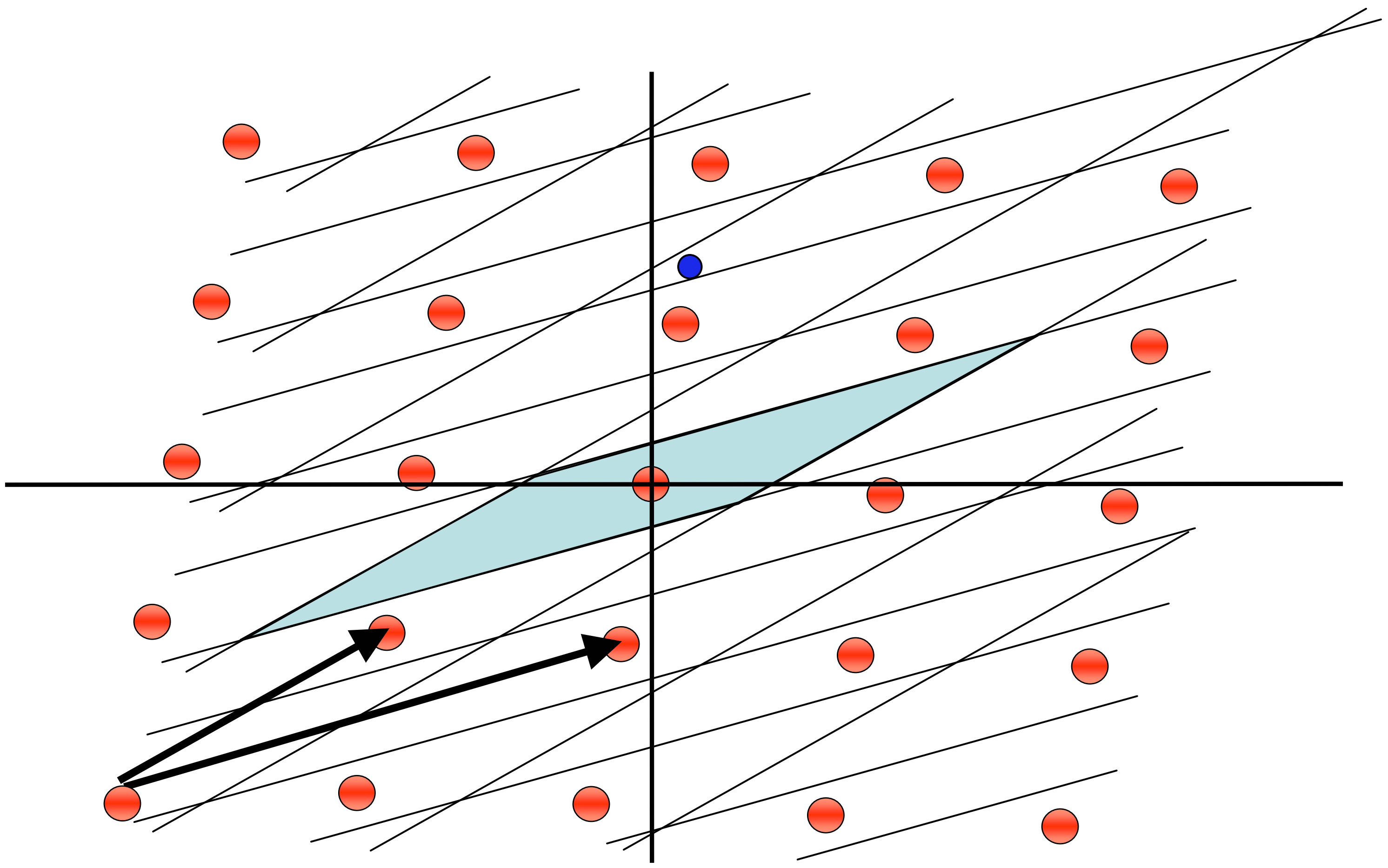


Good Basis

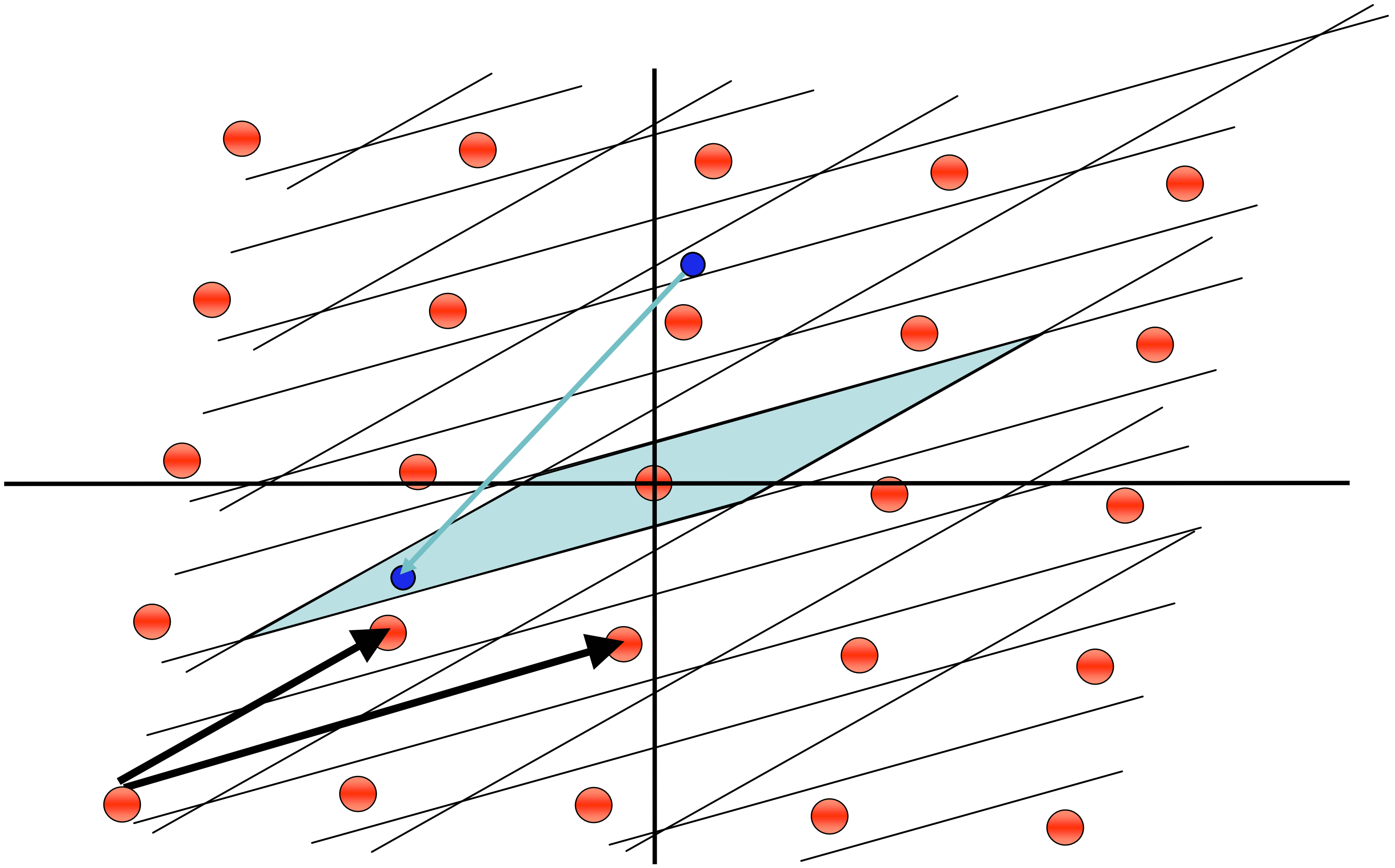


- Formula for reducing a basis modulo B = $\{v_1,v_2\}$: $t \bmod B = t - B\,[B^{-1} t]$

Bad Basis

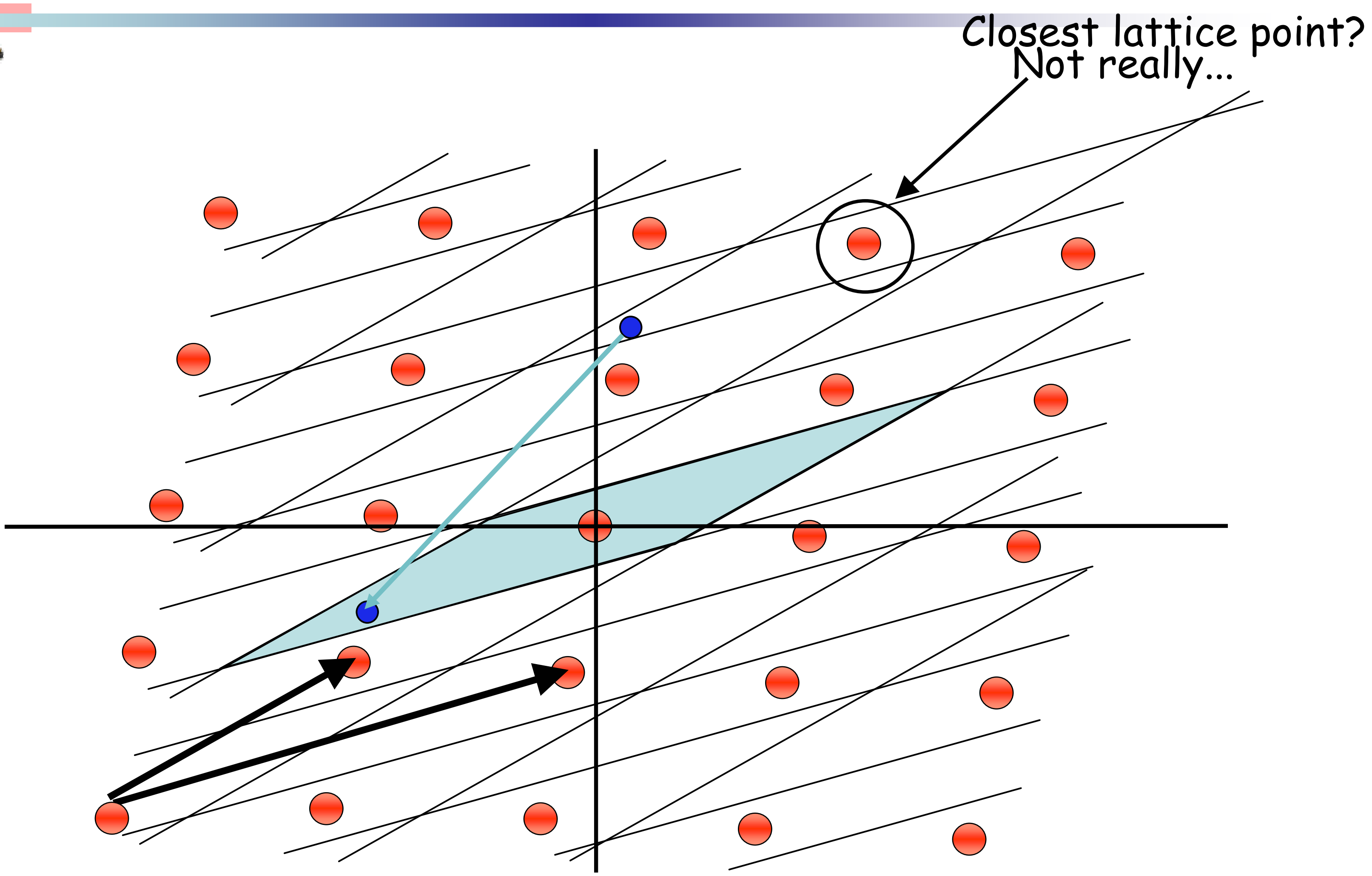


Bad Basis



- Formula for reducing a basis modulo B = $\{v_1,v_2\}$: $t \bmod B = t - B\,[B^{-1}\,t]$
- LLL 2^n-approximates the best basis.

Bad Basis



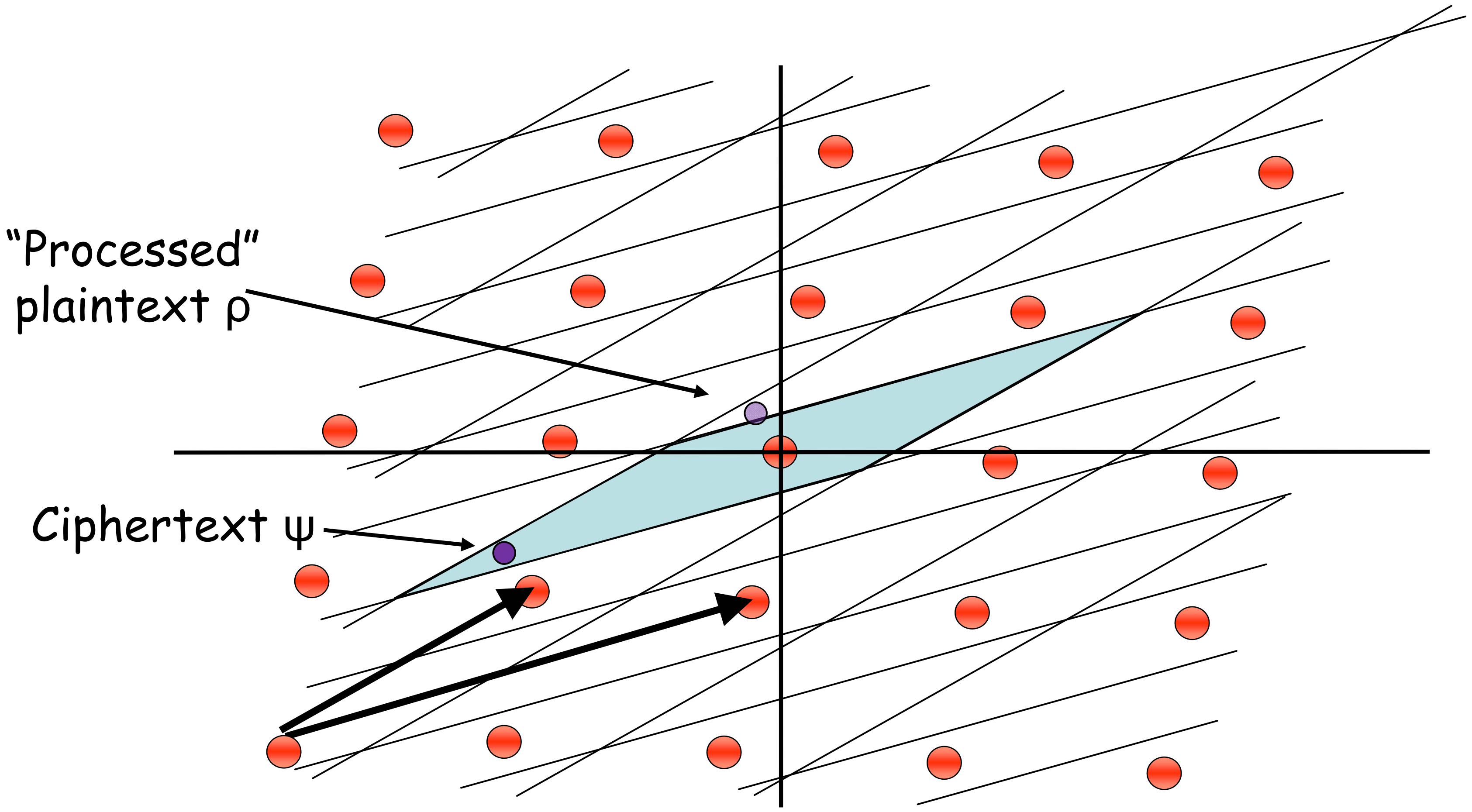
- Formula for reducing a basis modulo $B = \{v_1,v_2\}$: $t \bmod B = t - B\,[B^{-1}\,t]$
- LLL 2^n-approximates the best basis.

How Do We Encrypt Using Lattices?

- Ideas:
  - Close / Far: Ciphertext for 0 is close to a lattice point, and a ciphertext for 1 is far.
  - Odd / Even:
    - Encryption of 0: vector that differs from closest lattice point by an "even" vector.
    - Encryption of 1: vector that differs from closest lattice point by an "odd" vector.

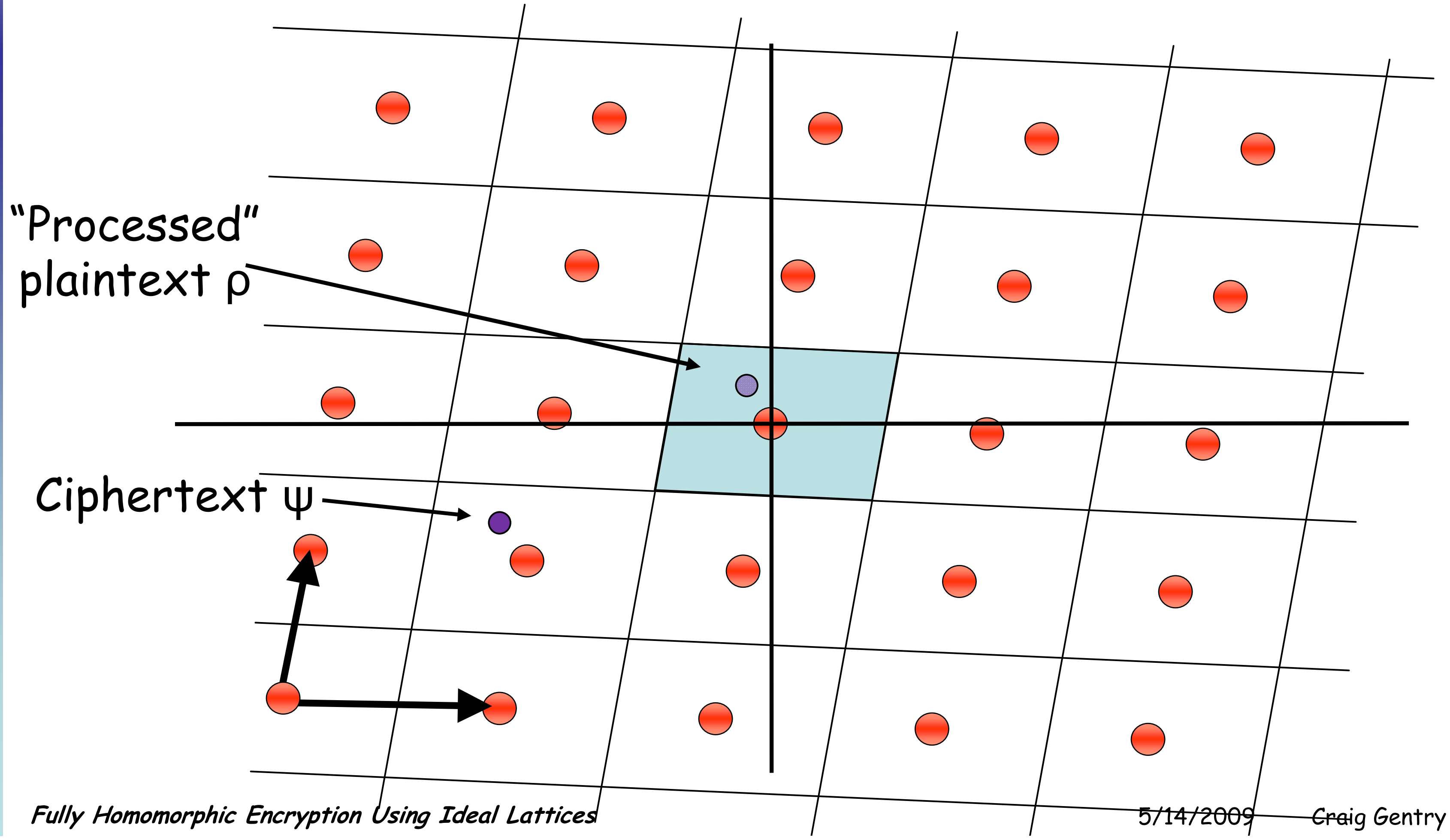
A Rough Lattice-Based Encryption Scheme

- Encryption: $\psi \leftarrow \rho \bmod B_{pk}$ (public basis)



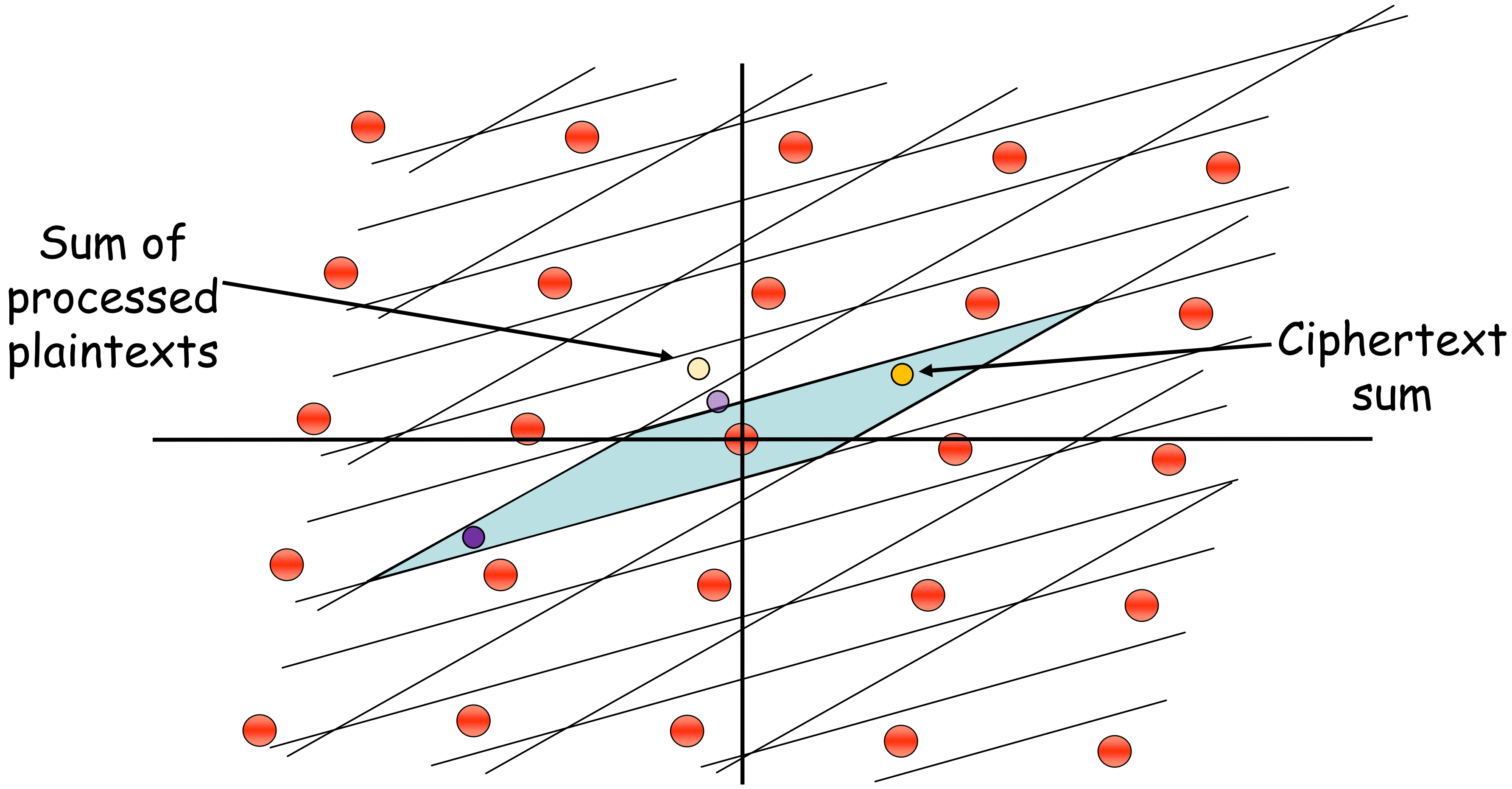
A Rough Lattice-Based Encryption Scheme

- Encryption: $\psi \leftarrow \rho \bmod B_{pk}$ (public basis)
- Decryption: $\rho \leftarrow \psi \bmod B_{sk}$ (secret basis) $= \psi - B_{sk}\,[B_{sk}^{-1}\,\psi]$



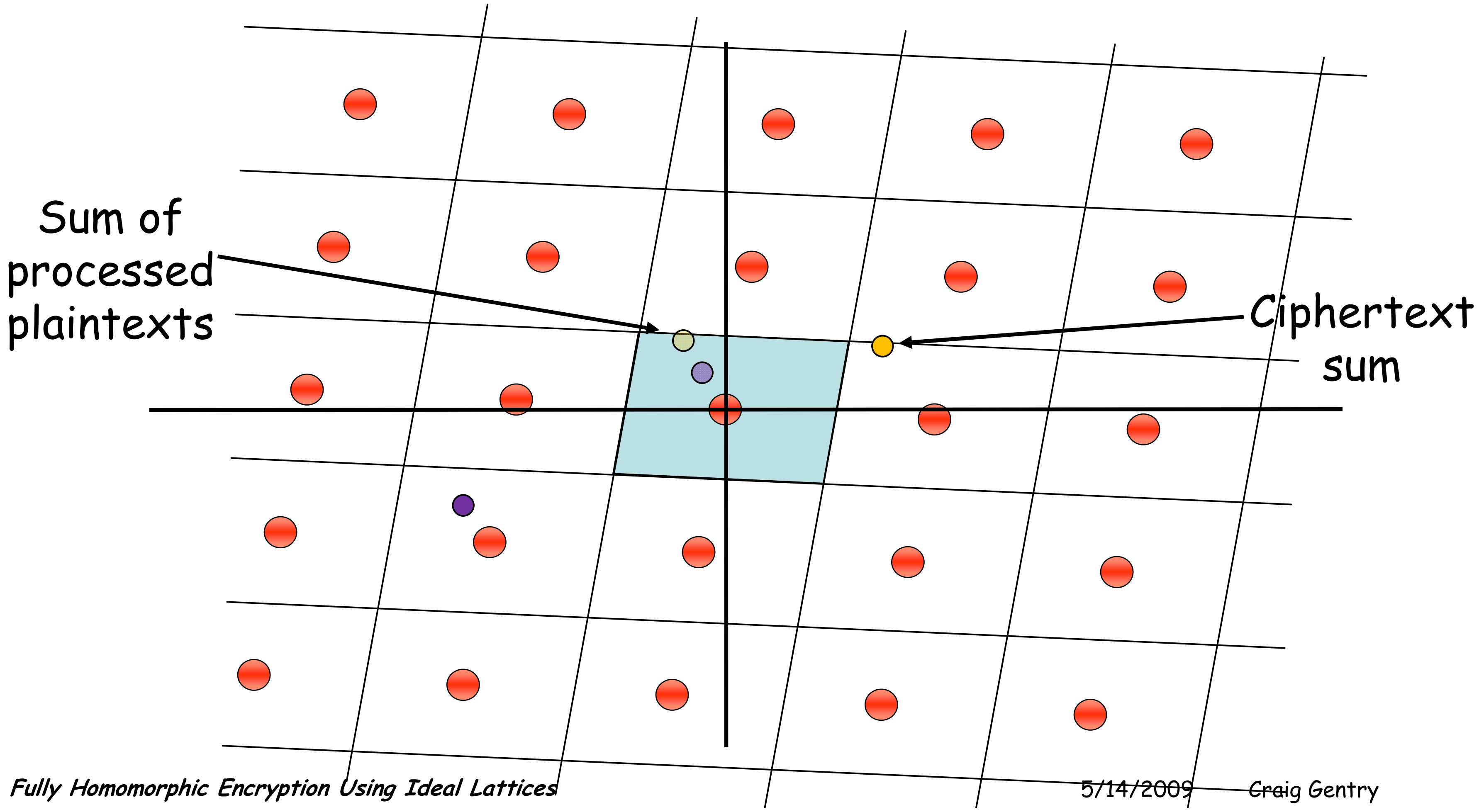
What if we add ciphertext vectors?

- Encryption: $\psi \leftarrow \rho \bmod B_{pk}$ (public basis)



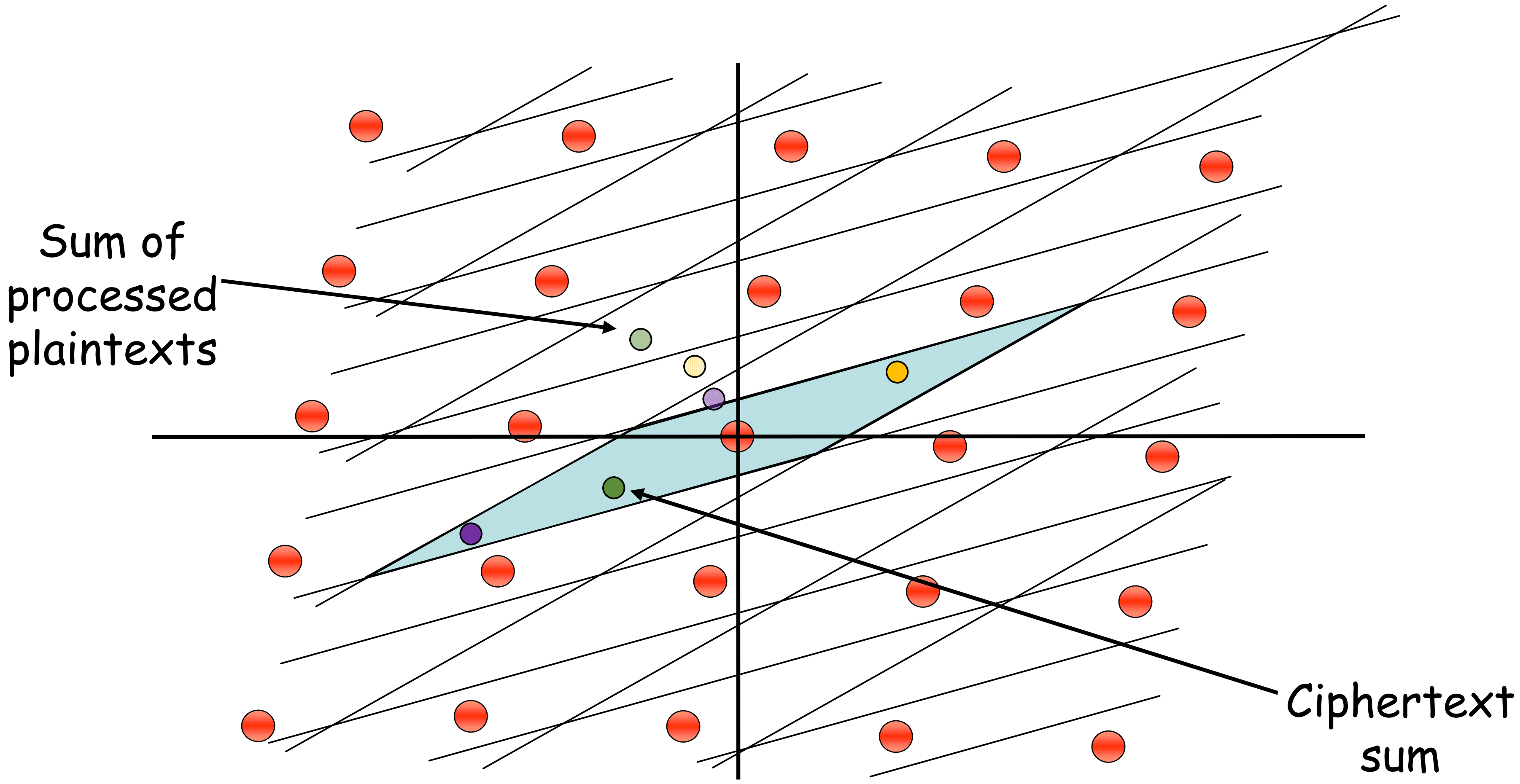
What if we add ciphertext vectors?

- Encryption: $\psi \leftarrow \rho \bmod B_{pk}$ (public basis)
- Decryption: $\rho \leftarrow \psi \bmod B_{sk}$ (secret basis) $= \psi - B_{sk}\,[B_{sk}^{-1}\,\psi]$



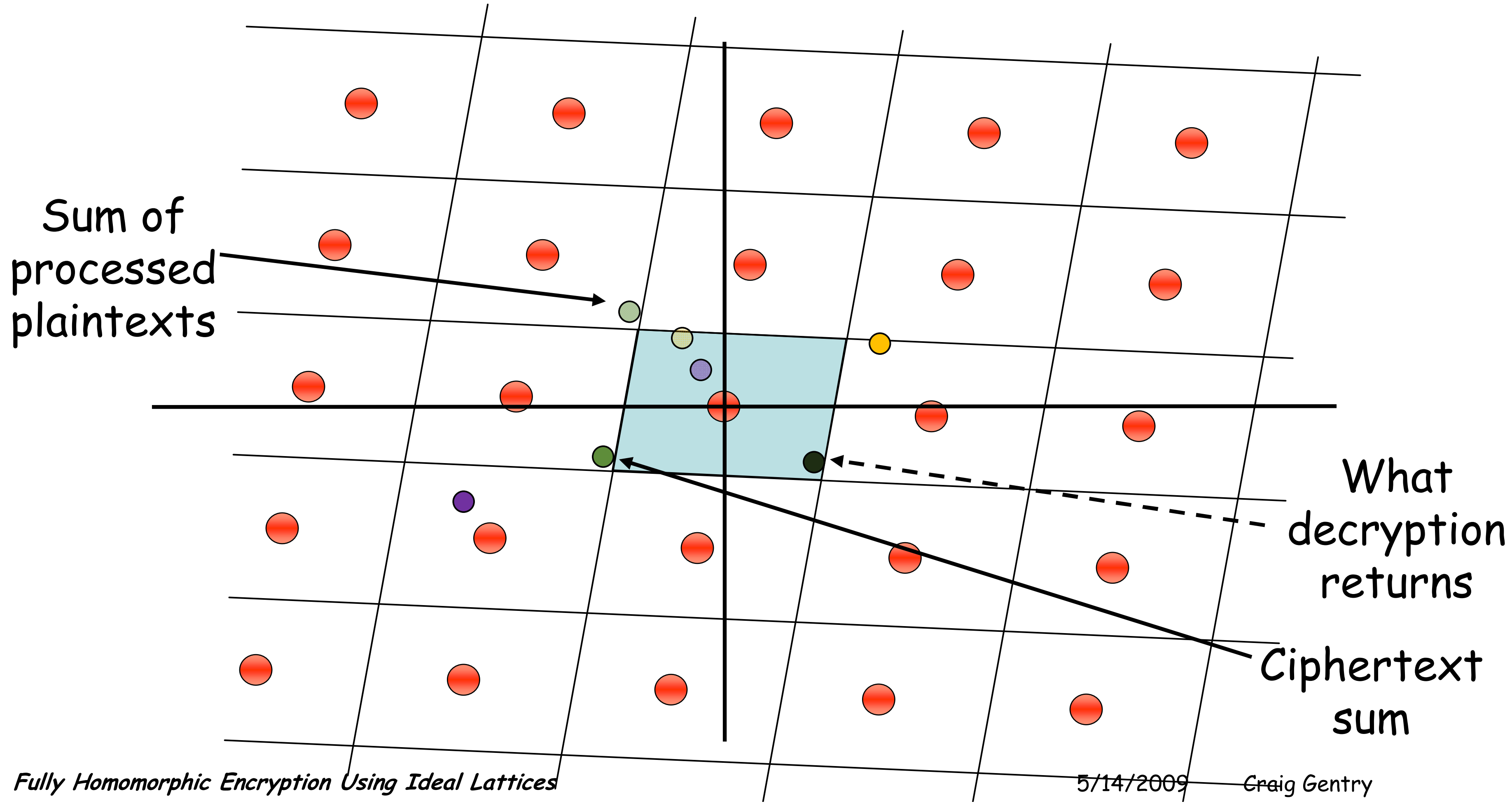
What if we add ciphertext vectors?

- Encryption: $\psi \leftarrow \rho \bmod B_{pk}$ (public basis)



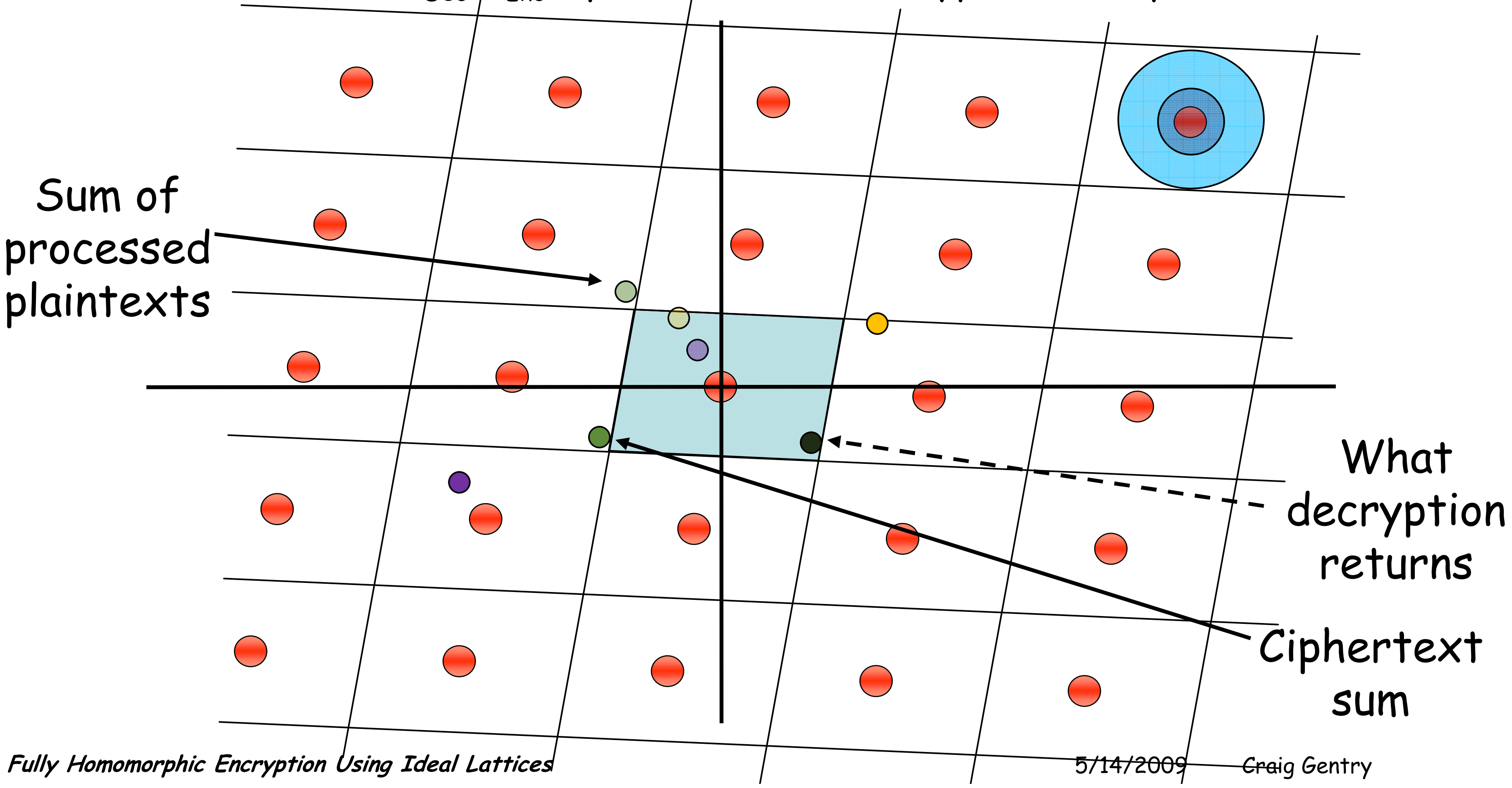
What if we add ciphertext vectors?

- Encryption: $\psi \leftarrow \rho \bmod B_{pk}$ (public basis)
- Decryption: $\rho \leftarrow \psi \bmod B_{sk}$ (secret basis) = $\psi - B_{sk}\,[B_{sk}^{-1}\,\psi]$



How many ciphertexts can we add?

- Suppose a sphere of radius r_{Dec} is in private parallelepiped.
- Suppose a processed plaintext is in $B(r_{Enc})$.
- We can add r_{Dec}/r_{Enc} ciphertexts, and decrypt correctly.



How many ciphertexts can we add?

- Fortunately, r_{Dec}/r_{Enc} can be huge – e.g., $2^{\sqrt{n}}$ – and still secure.
- LLL can find closest L-vector to t when

  $$\lambda_1(L)/\text{dist}(L,t) \quad > \quad 2^n$$

  where $\lambda_1(L)$ is the shortest nonzero vector in L.
- r_{Dec}: can as large as $\lambda_1(L)$, up to a small (poly(n)) factor.
- r_{Enc}: can be very small, as long as:
  - $\lambda_1(L)/r_{Enc}$ is not so large that LLL breaks security ($2^{\sqrt{n}}$ OK)
  - There is enough min-entropy in $B(r_{Enc})$, roughly speaking.
- Overall, r_{Dec}/r_{Enc} can be about $2^{\sqrt{n}}$.

How Can We Multiply Ciphertexts?

- Ideas:
  - Tensor Product: Would lead to huge ciphertexts
  - Use rings instead of (additive) groups: Good idea!

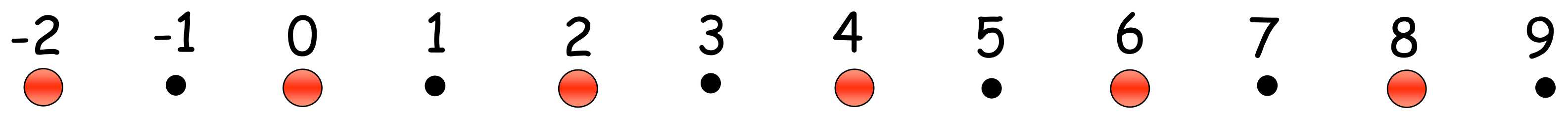
Ideal Lattices

What is an "ideal"?

A subset J of a ring R that is closed under "+", and also closed under "×" *with R*.

What is an "ideal lattice"? One object, both an ideal and a lattice

- Example: Z (integers) is a ring. (2), the even integers, is an ideal.



Ideal Lattices

What is an "ideal"?

A subset J of a ring R that is closed under "+", and also closed under "×" *with R.*

What is an "ideal lattice"? One object, both an ideal and a lattice

- Example: Z[x]/(f(x)) is a polynomial ring, f(x) monic, deg(f) = n.
- (a(x)) is an ideal { a(x)b(x) mod f(x) : b(x) in R }. Lattice basis below:

$a(x)$
$x \cdot a(x) \bmod f(x)$
…
$x^{n-1} \cdot a(x) \bmod f(x)$

a_0	a_1	a_2	…	a_{n-1}
$-a_{n-1}f_0$	$a_0-a_{n-1}f_1$	$a_1-a_{n-1}f_2$	…	$a_{n-2}-a_{n-1}f_{n-1}$
…				
…				

Ideal Lattice Scheme: High-Level

Background: CTs live in ring $R = Z[x]/f(x)$, where $\deg(f) = n$.
CTs can be added as vectors and multiplied as ring elements.

Ciphertext form: $m \quad + \quad 2\cdot v \quad + \quad j$

message in {0,1}

Random short even vector

Random vector from public key ideal J

Multiplication: $(m_1 + 2v_1 + j_1)\,(m_2 + 2v_2 + j_2)$

$= m_1 \times m_2 + 2(m_1v_2 + m_2v_1 + 2v_1v_2) + (\text{something in } J)$

Ideal Lattice Scheme: More Concretely

- Parameters: Ring $R = Z[x]/(f(x))$, basis B_I of "small" ideal lattice I. Radii r_{Dec} and r_{Enc} as before. The operations "+" and "×" are in R.
- KeyGen: Output "good" and "bad" bases (B_{sk}, B_{pk}) of a "big" ideal lattice J, which is relatively prime to I – i.e., I + J = R. Plaintext space: the cosets of I.
- Encrypt(B_{pk}, m): Set $m' \leftarrow^{R} (m+I) \cap B(r_{Enc})$. Set $c \leftarrow m' \bmod B_{pk}$.
- Decrypt(B_{sk}, c): Output $(c \bmod B_{sk}) \bmod B_I \rightarrow m$
- Add(B_{pk}, c_1, c_2): Output $c \leftarrow c_1 + c_2 \bmod B_{pk}$
- Mult(B_{pk}, c_1, c_2): Output $c \leftarrow c_1 \times c_2 \bmod B_{pk}$, which is in $m_1' \times m_2' + J$

The NTRU encryption scheme uses a similar approach with 2 relatively prime ideals.

Ideal Lattice Scheme: Correctness

- Parameters: Ring $R = Z[x]/(f(x))$, basis B_I of "small" ideal lattice I. Radii r_{Dec} and r_{Enc} as before. The operations "+" and "×" are in R.
- KeyGen: Output "good" and "bad" bases (B_{sk}, B_{pk}) of a "big" ideal lattice J, which is relatively prime to I – i.e., $I + J = R$. Plaintext space: the cosets of I.
- Encrypt(B_{pk}, m): Set $m' \xleftarrow{R} (m+I) \cap B(r_{Enc})$. Set $c \leftarrow m' \bmod B_{pk}$.
- Decrypt(B_{sk}, c): Output $(c \bmod B_{sk}) \bmod B_I \rightarrow m$
- Add(B_{pk}, c_1, c_2): Output $c \leftarrow c_1 + c_2 \bmod B_{pk}$
- Mult(B_{pk}, c_1, c_2): Output $c \leftarrow c_1 \times c_2 \bmod B_{pk}$, which is in $m_1' \times m_2' + J$

Correctness: Decryption works on Add(B_{pk}, c_1, c_2) if $m'_1+m'_2$ is in the B_{sk} parallelepiped.

Ideal Lattice Scheme: Correctness

- Parameters: Ring $R = Z[x]/(f(x))$, basis B_I of "small" ideal lattice I. Radii r_{Dec} and r_{Enc} as before. The operations "+" and "×" are in R.
- KeyGen: Output "good" and "bad" bases (B_{sk}, B_{pk}) of a "big" ideal lattice J, which is relatively prime to I – i.e., $I + J = R$. Plaintext space: the cosets of I.
- Encrypt(B_{pk}, m): Set $m' \xleftarrow{R} (m+I) \cap B(r_{Enc})$. Set $c \leftarrow m' \bmod B_{pk}$.
- Decrypt(B_{sk}, c): Output $(c \bmod B_{sk}) \bmod B_I \rightarrow m$
- Add(B_{pk}, c_1, c_2): Output $c \leftarrow c_1 + c_2 \bmod B_{pk}$
- Mult(B_{pk}, c_1, c_2): Output $c \leftarrow c_1 \times c_2 \bmod B_{pk}$, which is in $m_1' \times m_2' + J$

Correctness: Decryption works on Mult(B_{pk}, c_1, c_2) if $m'_1 \times m'_2$ is in the B_{sk} parallelepiped.

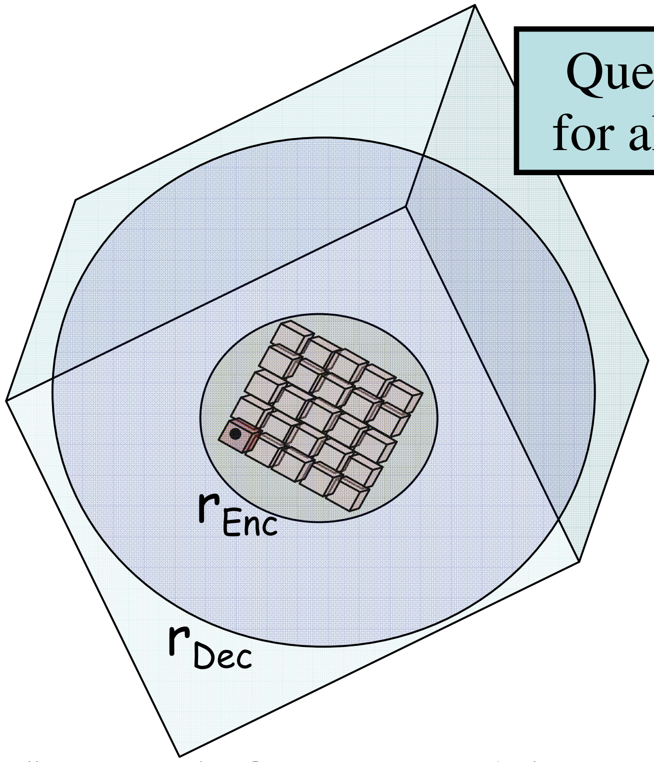
Ideal Lattice Scheme: Correctness

- Parameters: Ring $R = Z[x]/(f(x))$, basis B_I of "small" ideal lattice I. Radii r_{Dec} and r_{Enc} as before. The operations "+" and "×" are in R.
- KeyGen: Output "good" and "bad" bases (B_{sk}, B_{pk}) of a "big" ideal lattice J, which is relatively prime to I – i.e., I + J = R. Plaintext space: the cosets of I.
- Encrypt(B_{pk}, m): Set $m' \overset{R}{\leftarrow} (m+I) \cap B(r_{Enc})$. Set $c \leftarrow m' \bmod B_{pk}$.
- Decrypt(B_{sk}, c): Output $(c \bmod B_{sk}) \bmod B_I \rightarrow m$
- Add(B_{pk}, c_1, c_2): Output $c \leftarrow c_1 + c_2 \bmod B_{pk}$
- Mult(B_{pk}, c_1, c_2): Output $c \leftarrow c_1 \times c_2 \bmod B_{pk}$, which is in $m_1' \times m_2' + J$

Correctness: Correct for set S of circuits if $C(m'_1, \ldots, m'_t)$ is *always* in the B_{sk} parallelepiped..

Analyzing the Evaluative Capacity Geometrically

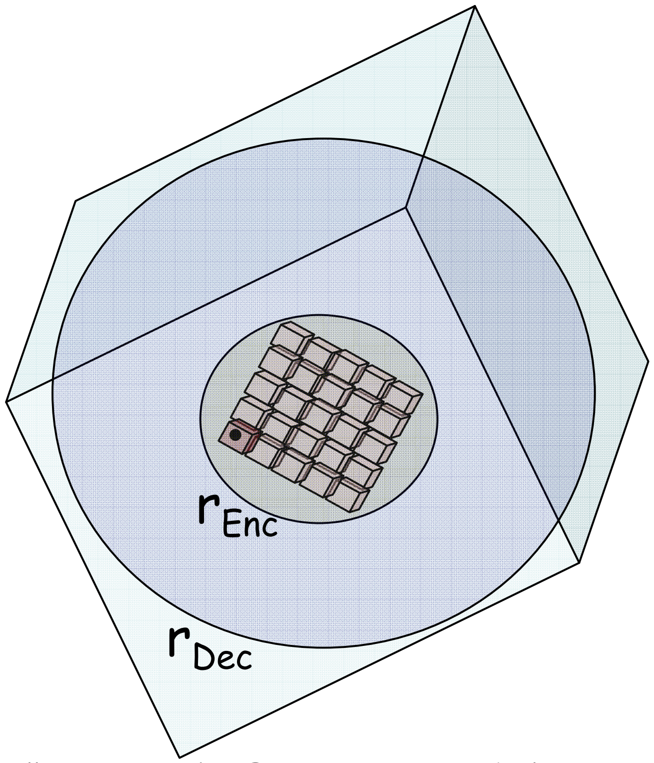
Correctness: Correct for set S of circuits if $C(m'_1, \ldots, m'_t)$ is *always* in the B_{sk} parallelepiped.



Question: for what arithmetic circuits C does this hold: for all $(x_1, \ldots, x_t)$ in $B(r_{Enc})^t$, $C(x_1, \ldots, x_t)$ is inside $B(r_{Dec})$

Analyzing the Evaluative Capacity Geometrically

Question: for what arithmetic circuits C does this hold: for all $(x_1, ..., x_t)$ in $B(r_{Enc})^t$, $C(x_1, ..., x_t)$ is inside $B(r_{Dec})$



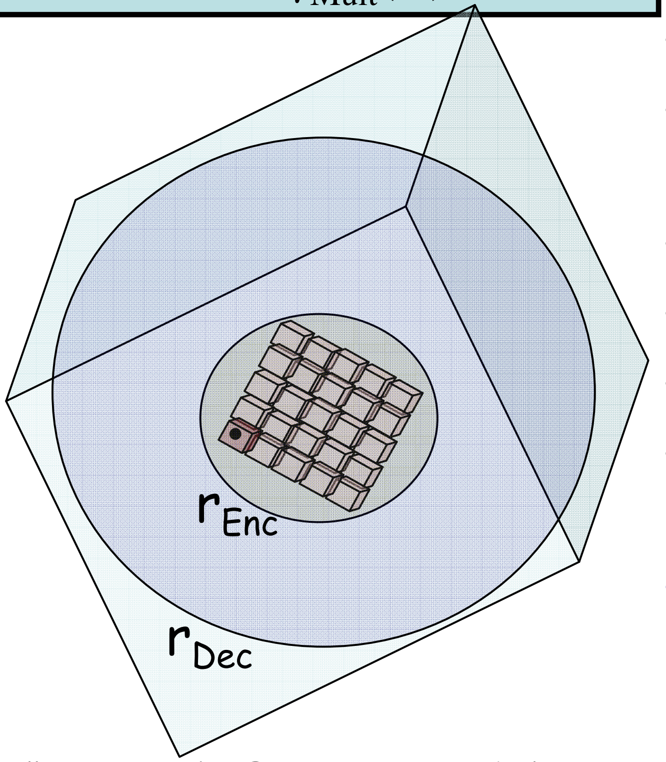
- Add operations: $|u+v| \leq |u| + |v|$ (triangle inequality)
- Mult operations: $|u \times v| \leq \gamma_{Mult}(R) \cdot |u| \cdot |v|$ for some factor $\gamma_{Mult}(R)$ that depends on the ring R, and which can be poly(n).
- Add vs. Mult:
  - Add causes much less expansion than Mult.
  - Constant fan-in Mult is as bad as poly(n) fan-in Add.

Analyzing the Evaluative Capacity Geometrically

Question: for what arithmetic circuits C does this hold: for all $(x_1, ..., x_t)$ in $B(r_{Enc})^t$, $C(x_1, ..., x_t)$ is inside $B(r_{Dec})$

Add: $|u+v| \leq |u| + |v|$

Mult: $|u \times v| \leq \gamma_{Mult}(R) \cdot |u| \cdot |v|$



How much depth can we get?

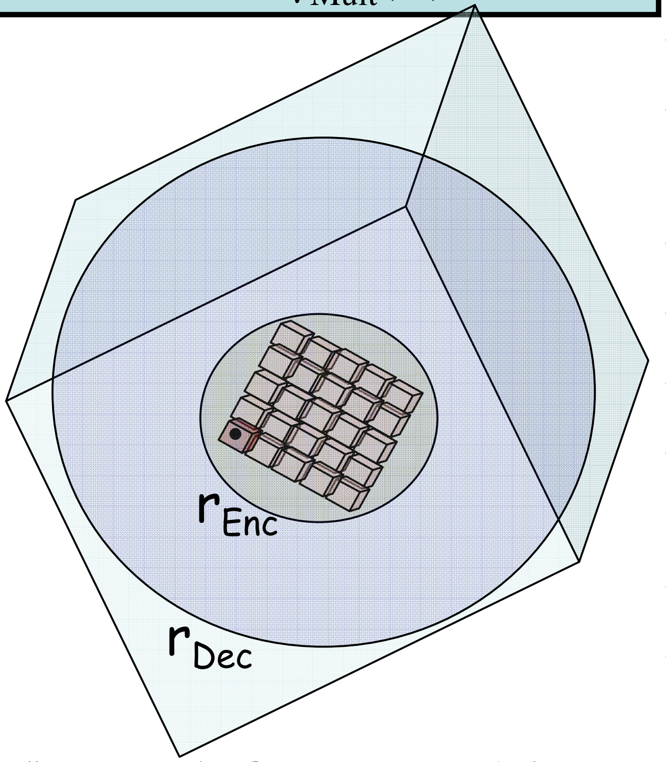
- Let C be a fan-in-2, depth d arithmetic circuit
- Let r_i be the max radius associated to a gate in C at level i, when $r_d = r_{Enc}$.
- $r_i \leq \gamma_{Mult}(R) \cdot r_{i+1}^2$
- Then, $r_0 \leq (\gamma_{Mult}(R) \cdot r_d)^{2^d}$.
- $r_0 \leq r_{Dec}$ if $d \leq \log \log r_{Dec} - \log \log (\gamma_{Mult}(R) \cdot r_{Enc})$
- E.g., $(c_1-c_2) \log n$ depth when $r_{Dec} = 2^{n^{c1}}$ and $\gamma_{Mult}(R) \cdot r_{Enc} = 2^{n^{c2}}$.
- Bottom line: We get about log n depth.

Analyzing the Evaluative Capacity Geometrically

Question: for what arithmetic circuits C does this hold: for all $(x_1, ..., x_t)$ in $B(r_{Enc})^t$, $C(x_1, ..., x_t)$ is inside $B(r_{Dec})$

Add: $|u+v| \leq |u| + |v|$

Mult: $|u \times v| \leq \gamma_{Mult}(R) \cdot |u| \cdot |v|$



How much depth can we get?

- Let C be a fan-in-2, depth d arithmetic circuit
- Let r_i be the max radius associated to a gate in C at level i, when $r_d = r_{Enc}$.
- $r_i \leq \gamma_{Mult}(R) \cdot r_{i+1}^2$
- Then, $r_0 \leq (\gamma_{Mult}(R) \cdot r_d)^{2^d}$.
- $r_0 \leq r_{Dec}$ if $d \leq \log \log r_{Dec} - \log \log (\gamma_{Mult}(R) \cdot r_{Enc})$
- E.g., $(c_1 - c_2) \log n$ depth when $r_{Dec} = 2^{n^{c1}}$ and $\gamma_{Mult}(R) \cdot r_{Enc} = 2^{n^{c2}}$.
- Bottom line: We get about log n depth.
- Is this enough to bootstrap??

Homomorphic Decryption to "Refresh" Ciphertexts

- Intuition: When our ciphertext's "error vector" becomes to long, we want to "refresh" the ciphertext:
  - Get a new encryption of same plaintext with shorter error.
- How to do it?
  - Decrypt it, then encrypt again!
    - But this requires the secret key…

Homomorphic Decryption to “Refresh” Ciphertexts

- Intuition: When our ciphertext’s “error vector” becomes to long, we want to “refresh” the ciphertext:
    - Get a new encryption of same plaintext with shorter error.
- How to do it?
    - Decrypt it, then encrypt again!
        - But this requires the secret key…
    - Homomorphically decrypt it!!!

The Decryption Circuit of the Initial Scheme

$\text{Decrypt}(B_{sk}, \psi) = (\psi \bmod B_{sk}) \bmod B_I$

$= (\psi - B_{sk} \cdot [B_{sk}^{-1} \cdot \psi]) \bmod B_I$

Can simplify this to:

$\text{Decrypt}(v_{sk}, \psi) = (\psi - [(v_{sk})^{-1} \times \psi]) \bmod (2)$

$$\boxed{\text{Expensive Step: Computing } [(v_{sk})^{-1} \times \psi] \bmod (2)}$$

Another "tweak": Require ψ to be within $r_{Dec}/2$ of a lattice point.
Then, the coeffs of $(v_{sk})^{-1} \times \psi$ will be within $\frac{1}{4}$ of an integer.
Then, we need less precision to ensure correct rounding.

The Decryption Circuit of the Initial Scheme

Expensive Step: Computing $[(v_{sk})^{-1} \times \psi] \bmod (2)$

- Ring multiplication is like a bunch of parallel inner products
- Each inner product involves an addition of n terms, like this:

  1.1101… + 0.0101… + 0.1011… + 1.1010… + …

- We have to worry about *carry bits* -> have high degree in input.
- When vectors are n-dimensional, the shallowest circuit I know of has depth O(log n), and is heavy on the MULTs.

The Decryption Circuit of the Initial Scheme

Expensive Step: Computing $[(v_{sk})^{-1} \times \psi] \bmod 2$

1.1101… + 0.0101… + 0.1011… + 1.1010… + …

- When vectors are n-dimensional, the least complex circuit I know of has depth O(log n), and is heavy on the MULTs.
  - "3-for-2" trick: replaces 3 (binary) numbers with 2 numbers having the same sum.
  - $c \log_{3/2} n$ depth to get 2 numbers with same sum as n numbers.

0.1011… + 1.0111…

  - Normally, depth of adding 2 numbers is log in their bit-lengths
  - But, we can use fact that, for valid ciphertexts, $(v_{sk})^{-1} \times \psi$ is very close to an integer vector -> final sum is constant depth.

The Decryption Circuit of the Initial Scheme

- Bottom line: Decryption circuit is also O(log n), but for a larger constant than the depth we can Evaluate.
- Blargh…

Still Not Bad...

- Boneh-Goh-Nissim does quadratic formulas: arbitrary number of additions, but multiplication depth of 1.
- Our scheme:
    - Essentially arbitrary additions, but with log n multiplication depth.
    - Also, larger plaintext space.

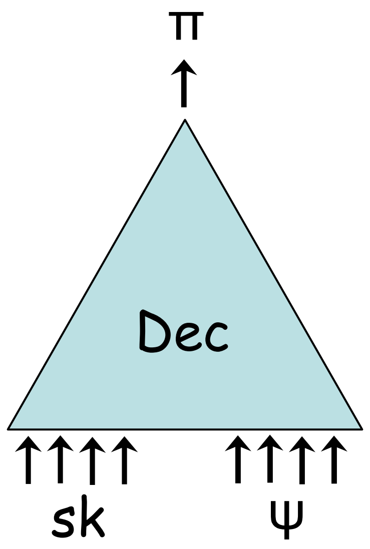
Security of the scheme

- We'll discuss this in more detail later if we have time...

Step 3: Squashing the Decryption Circuit

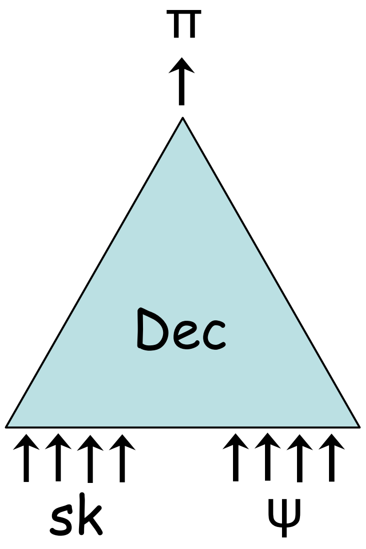
Abstractly, How Can We Lower the Decryption Complexity?

Old decryption algorithm



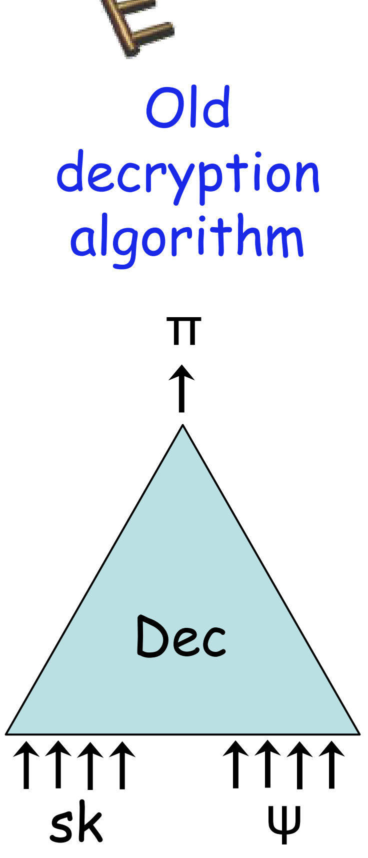
Abstractly, How Can We Lower the Decryption Complexity?

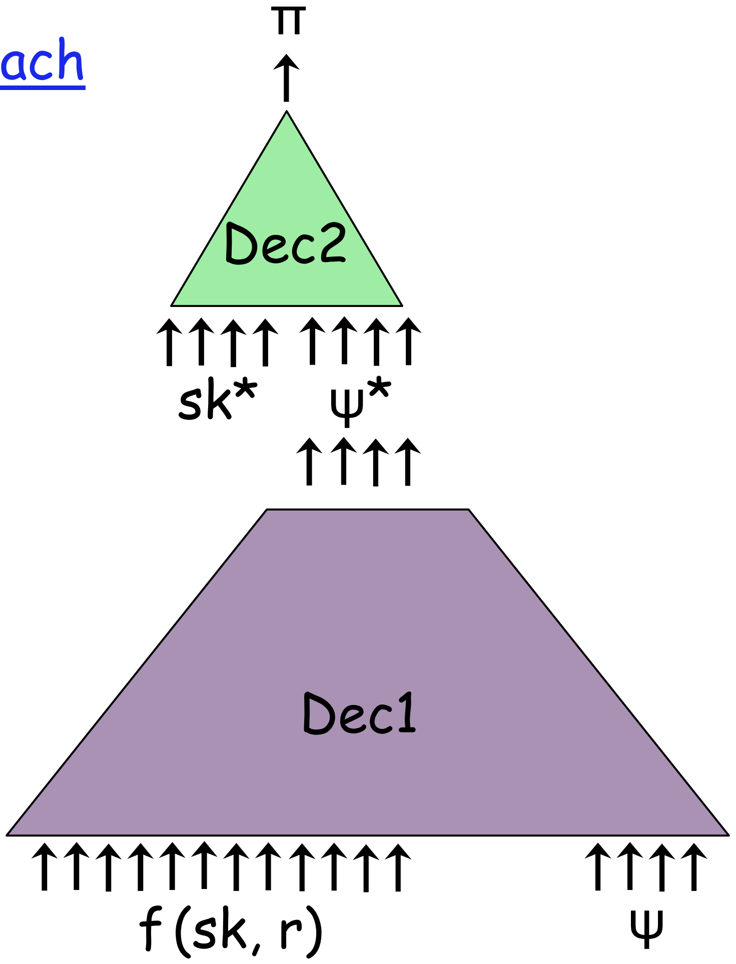
Old decryption algorithm



Crazy idea: The encrypter starts decryption, leaving less for the decrypter to do!

Abstractly, How Can We Lower the Decryption Complexity?

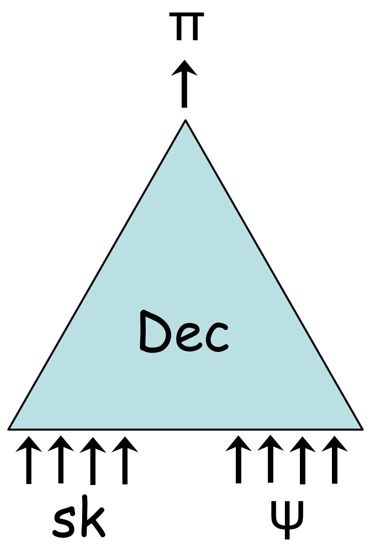




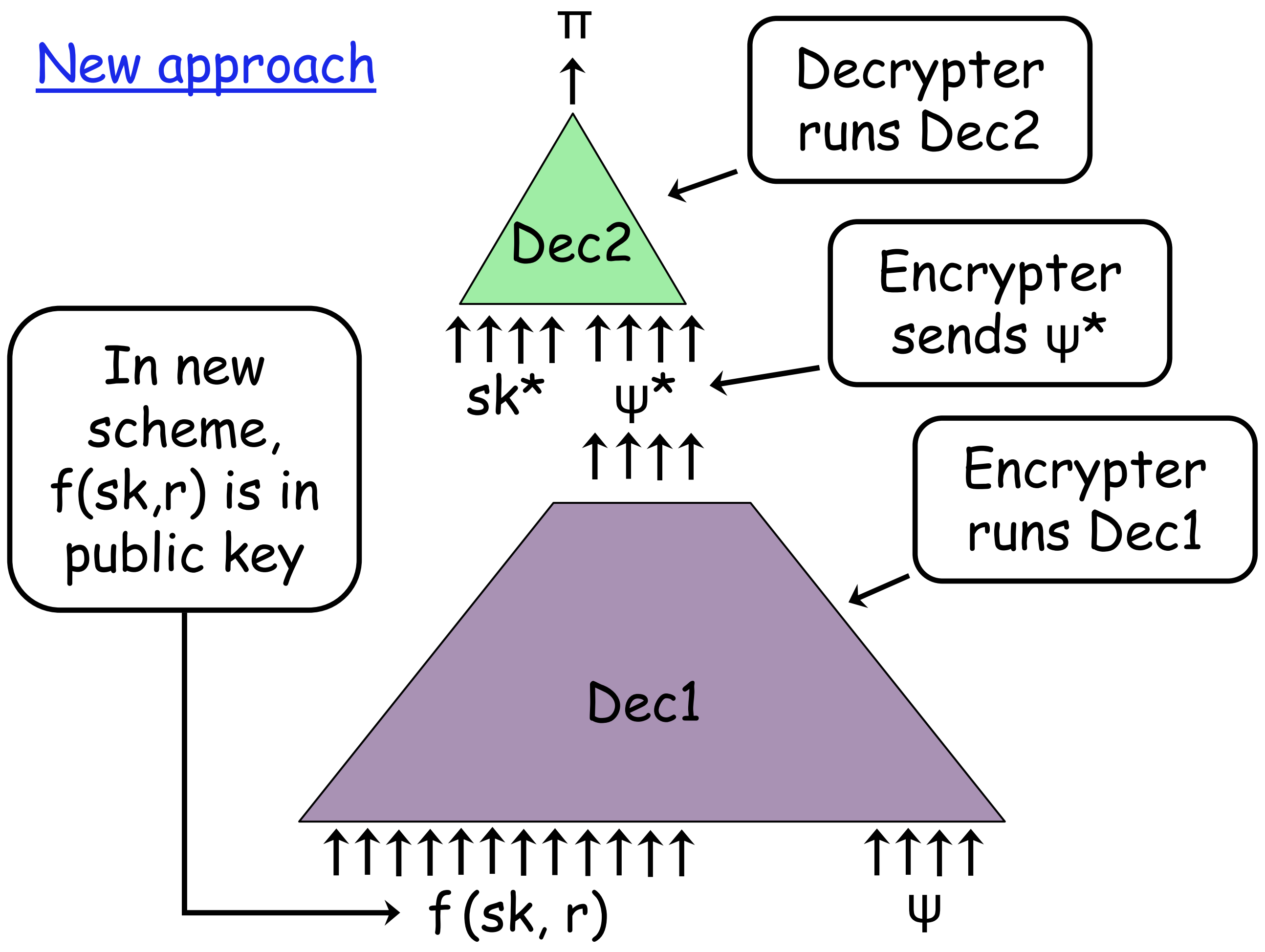
Crazy idea: The encrypter starts decryption, leaving less for the decrypter to do!

Abstractly, How Can We Lower the Decryption Complexity?

Old decryption algorithm

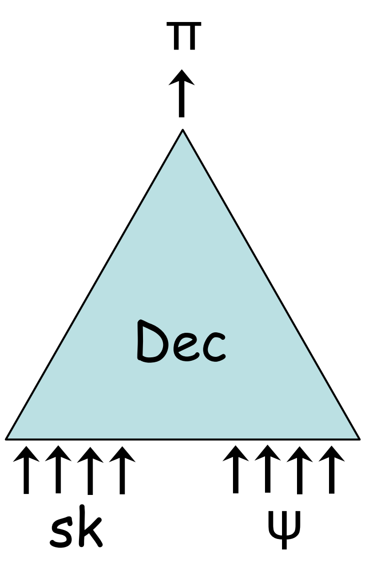


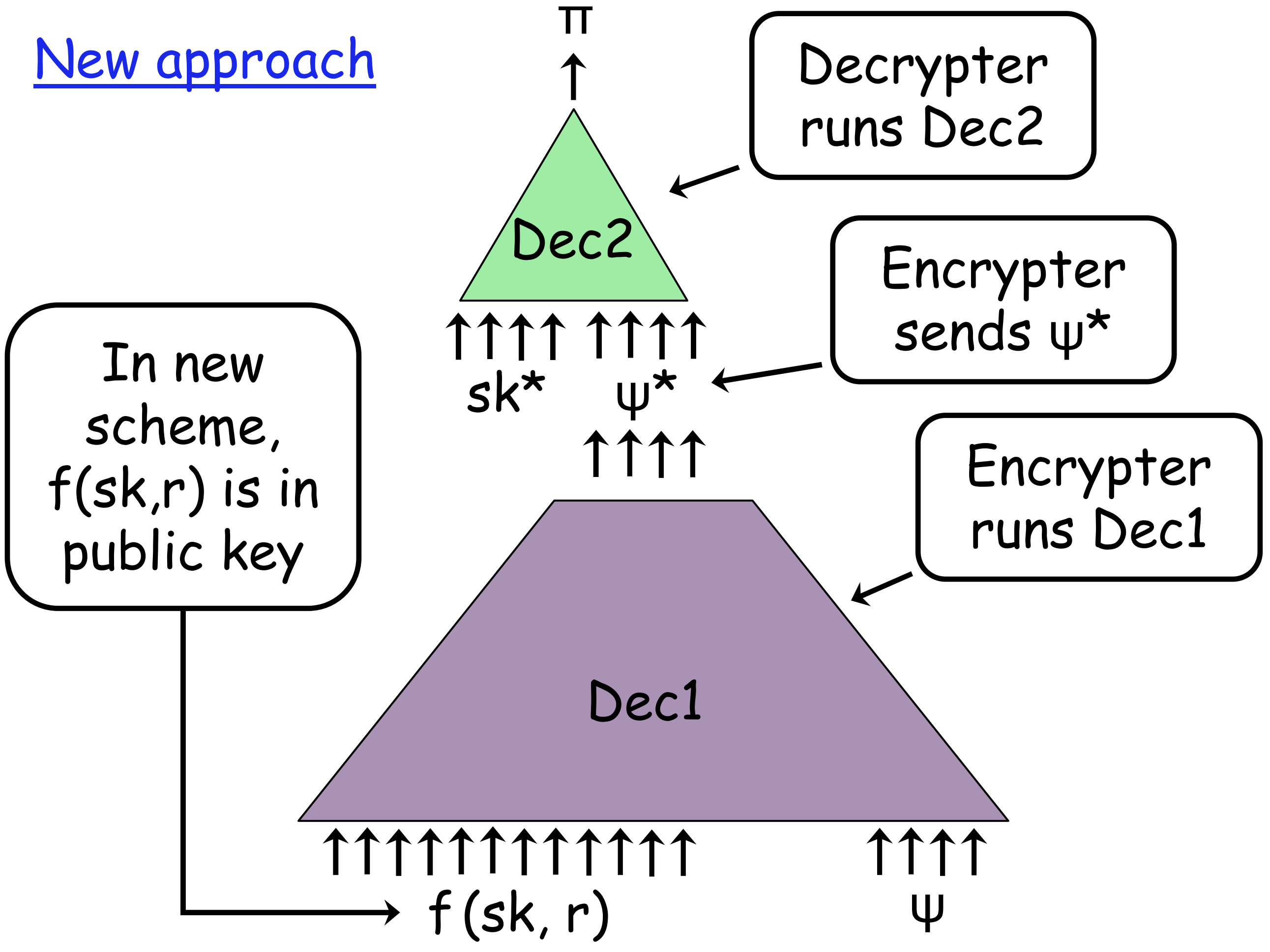
New approach



Crazy idea: The encrypter starts decryption, leaving less for the decrypter to do!

Abstractly, How Can We Lower the Decryption Complexity?

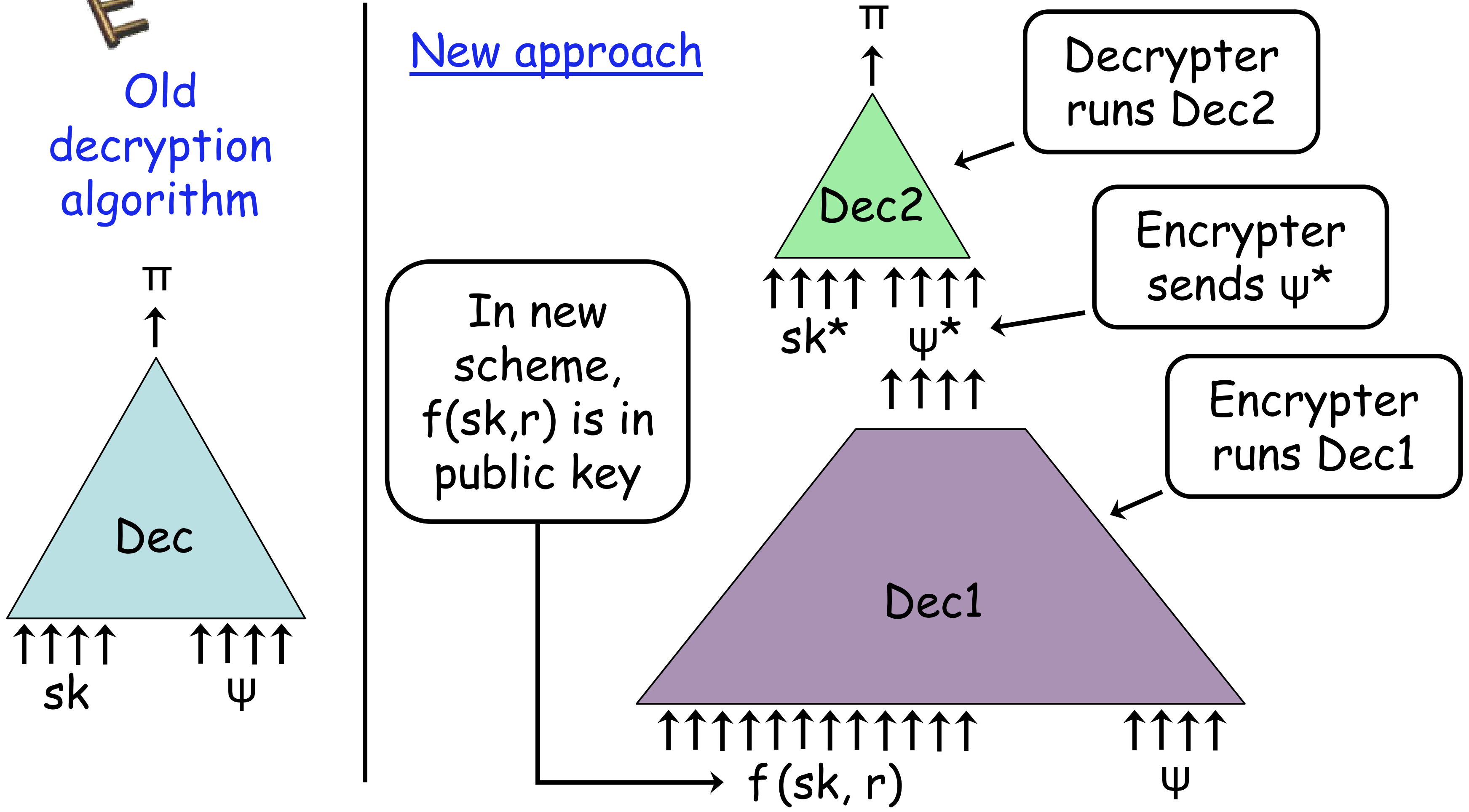




Dec2 should be less complex than Dec

(Dec1, Dec2) should work on any ψ that Dec works on

Abstractly, How Can We Lower the Decryption Complexity?



Still semantically secure if f(sk,r) is computationally indistinguishable from random given (pk, sk), but not sk*.

Concretely, How Does the Transformation Work?

$$\text{Decrypt}(v_{sk}, \psi) \quad = \quad (\psi - [(v_{sk})^{-1} \times \psi]) \bmod (2)$$

Expensive Step: Computing $[(v_{sk})^{-1} \times \psi] \bmod 2$

Remember the Old Circuit…

Expensive Step: Computing $[(v_{sk})^{-1} \times \psi]$ mod 2

1.1101… + 0.0101… + 0.1011… + 1.1010… + …

- Dominant computation: "3-for-2 trick" circuit of depth $c \log_{3/2} n$

Our New Circuit…

Expensive Step: Computing $[(v_{sk})^{-1} \times \psi] \bmod 2$

1.1101… + 0.0101… + 0.1011… + 1.1010… + …

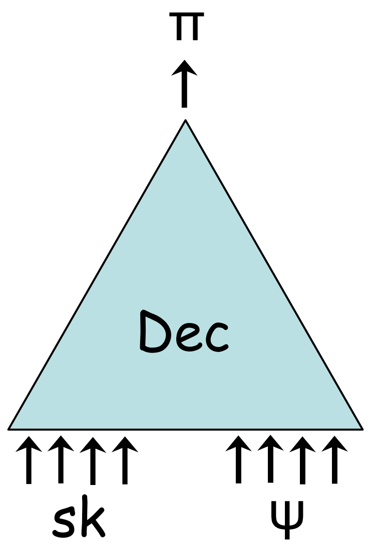
- Dominant computation: "3-for-2 trick" circuit of depth $c \log_{3/2} n$
- Goal: Use *less depth* to get 2 vectors

  (0.1011…, …, 1.0110…) + (1.0111…, …, 1.1000…)

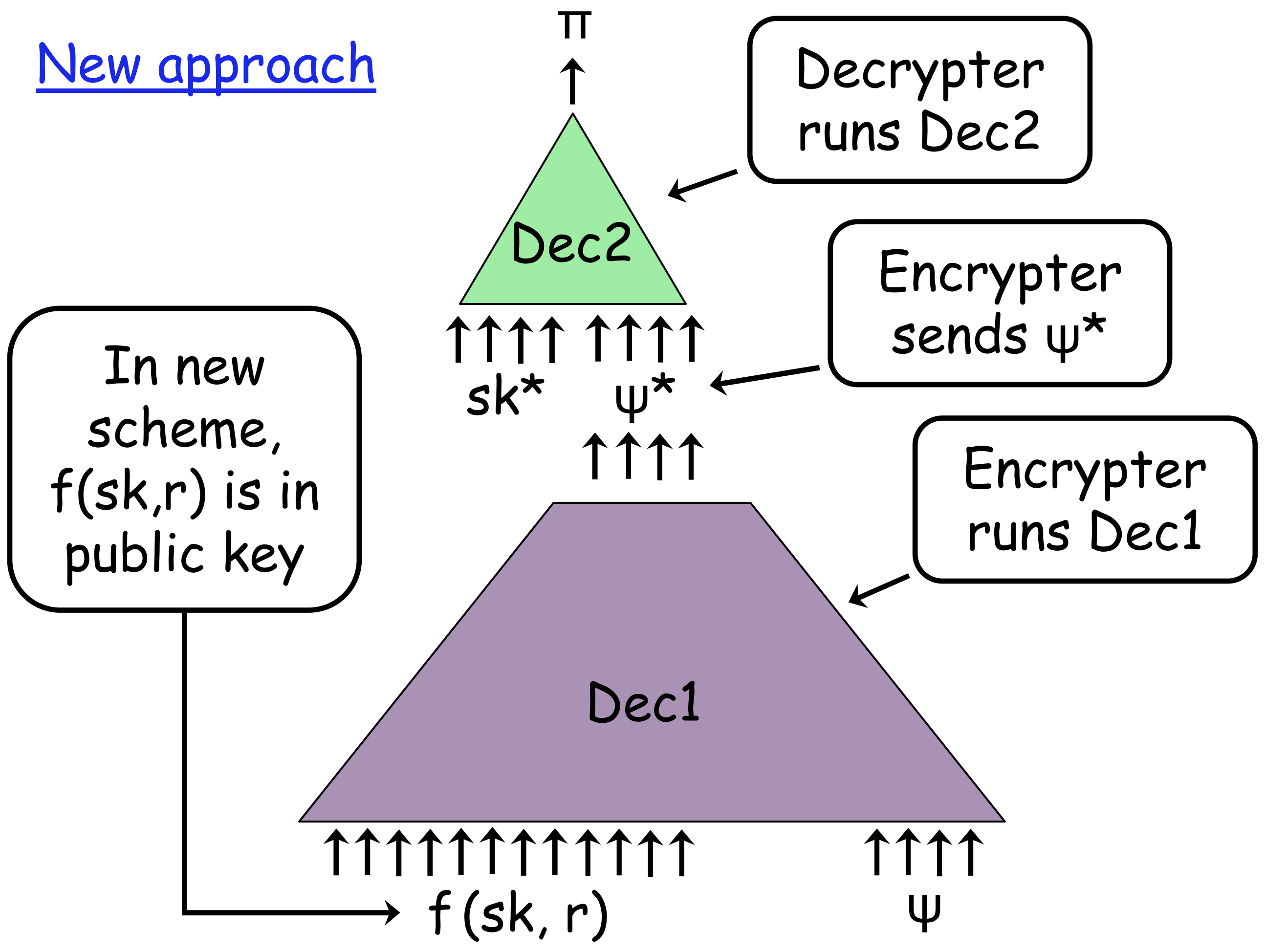
  whose sum is same (mod 2) as: $(v_{sk})^{-1} \times \psi$

- Strategy: Start with much fewer than n vectors in the first place!

Abstractly, How Can We Lower the Decryption Complexity?



New approach



Still semantically secure if f(sk,r) is computationally indistinguishable from random given (pk, sk), but not sk*.

Concretely, How Does the New Approach Work?

Expensive Step: Computing $[(v_{sk})^{-1} \times \psi]$ mod 2

What is the "hint" f(sk,r) that we put in the pub key?

- The Hint: a set S of vectors $\{w_i\}$ that has a *hidden subset* T of vectors $\{x_i\}$ whose sum is $(v_{sk})^{-1}$.
- $|S| = n^{\beta}$, $\beta > 1$. $|T| = \omega(1)$ and $o(n)$.
- Dec1: Encrypter sends ψ and

$$\psi^* = \{ c_i = w_i \times \psi \pmod 2 \} \quad \text{for all } w_i \text{ in } S$$

- Dec2: Decrypter sums up the |T| values that are "relevant." This takes c log |T| depth with 3-for-2 trick.

Concretely, How Does the New Approach Work?

- The Hint: a set S of vectors $\{w_i\}$ that has a *hidden subset* T of vectors $\{x_i\}$ whose sum is $(v_{sk})^{-1}$.
- $|S| = n^{\beta}$, $\beta > 1$. $|T| = \omega(1)$ and $o(n)$.
- Dec1: Encrypter sends ψ and

  $\psi^* = \{ c_i = w_i \times \psi \pmod 2 \}$ for all w_i in S
- Dec2: Decrypter sums up the $|T|$ vectors that are "relevant." This takes $c \log |T|$ depth with 3-for-2 trick.

In Dec2, how do we cheaply extract |T| vectors that are relevant?

- Decrypter's secret key sk* consists of $|T|$ 0/1-vectors $\{y_i\}$ of dimension $|S|$; each encodes 1 member of $|T|$.

y_1:	0	1	0	0	0	0	0
y_2:	0	0	1	0	0	0	0
y_3:	0	0	0	0	0	1	0

- For each i, it inner-products y_i with ψ^*.
- Key point: No carries to worry about in inner product -> We can use a high fan-in add gate (cheap).

Concretely, How Does the New Approach Work?

Expensive Step: Computing $[(v_{sk})^{-1} \times \psi] \bmod 2$

- Bottom line: Dec2 has about log |T| depth, $|T| = \omega(1)$ and $o(n)$.
- New Assumption: Given set S of vectors $\{w_i\}$ and vector v, decide whether there exists a low-weight subset $T = \{x_i\}$ with $v = \Sigma x_i$.
- Can pick |S| s.t. there will be many subsets of size, say, |S|/2 whose sum is v.
- Known attacks: Finding T takes time roughly $n^{|T|}$.
- To evaluate depth log |T|, original scheme needs $r_{Dec}/r_{Enc} \approx n^{\Theta(|T|)}$. This is also basically the approx factor of the lattice problem.
  - Known attacks: Takes time roughly $2^{n/|T|}$.
  - Optimal: Set $|T| \approx \sqrt{n}$.

Performance

- Well… a little slow.
- "Evaluating" a circuit homomorphically takes $\tilde{O}(k^7)$ computation per circuit gate if you want 2^k security against known attacks.
- … But a full exponentiation in RSA also takes $\tilde{O}(k^6)$; also, in ElGamal (using finite fields).

Open Problems

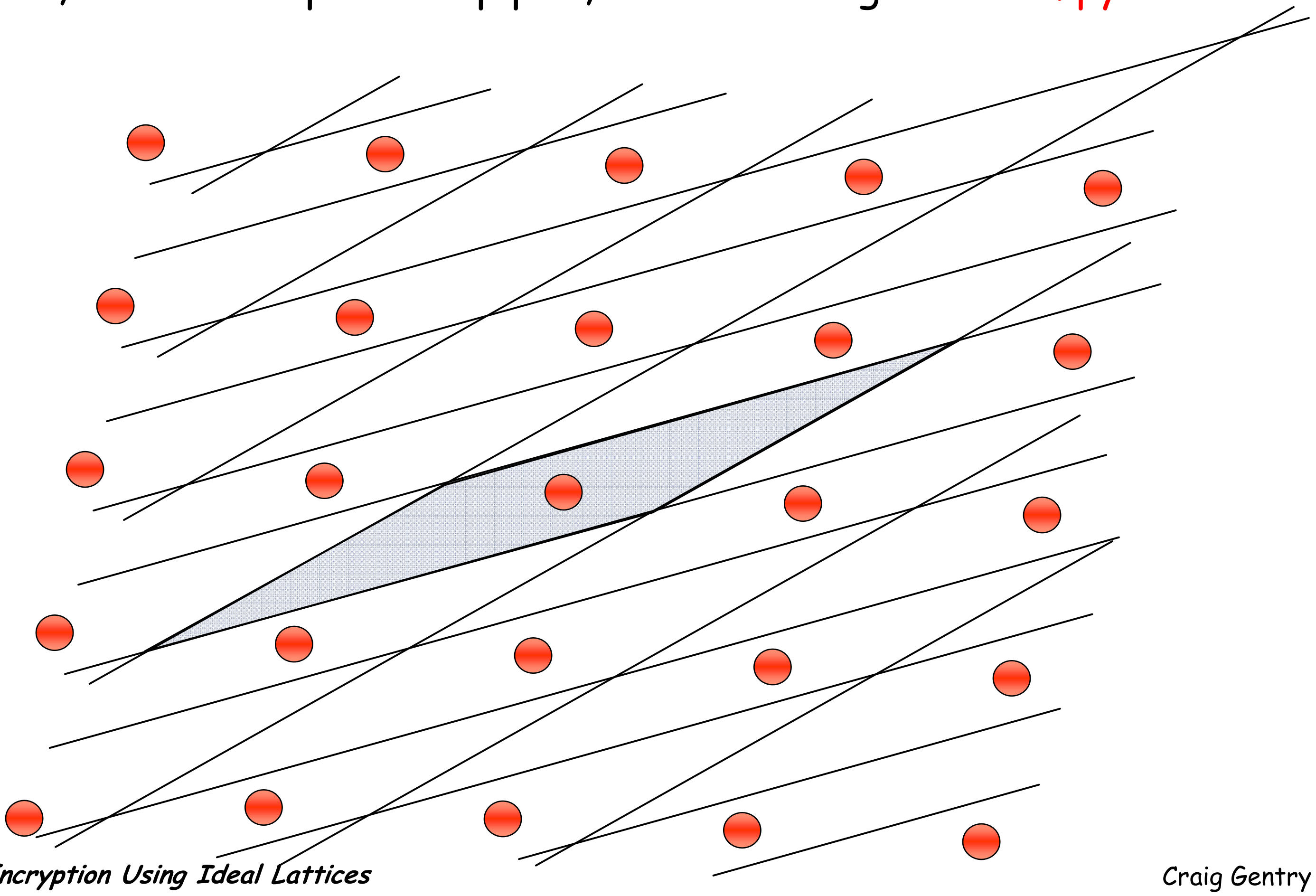
- CCA1 Security
- Improve efficiency
- System using linear codes (wouldn't be so surprising)
- System based on "conventional" crypto assumptions
- "Refreshing" a ciphertext without completely (homomorphically) decrypting it

Thank You! Questions?

Security of the Initial Ideal Lattice Scheme

Distributional CVP: Generate basis B_{pk} for ideal lattice J using KeyGen. Set bit b.

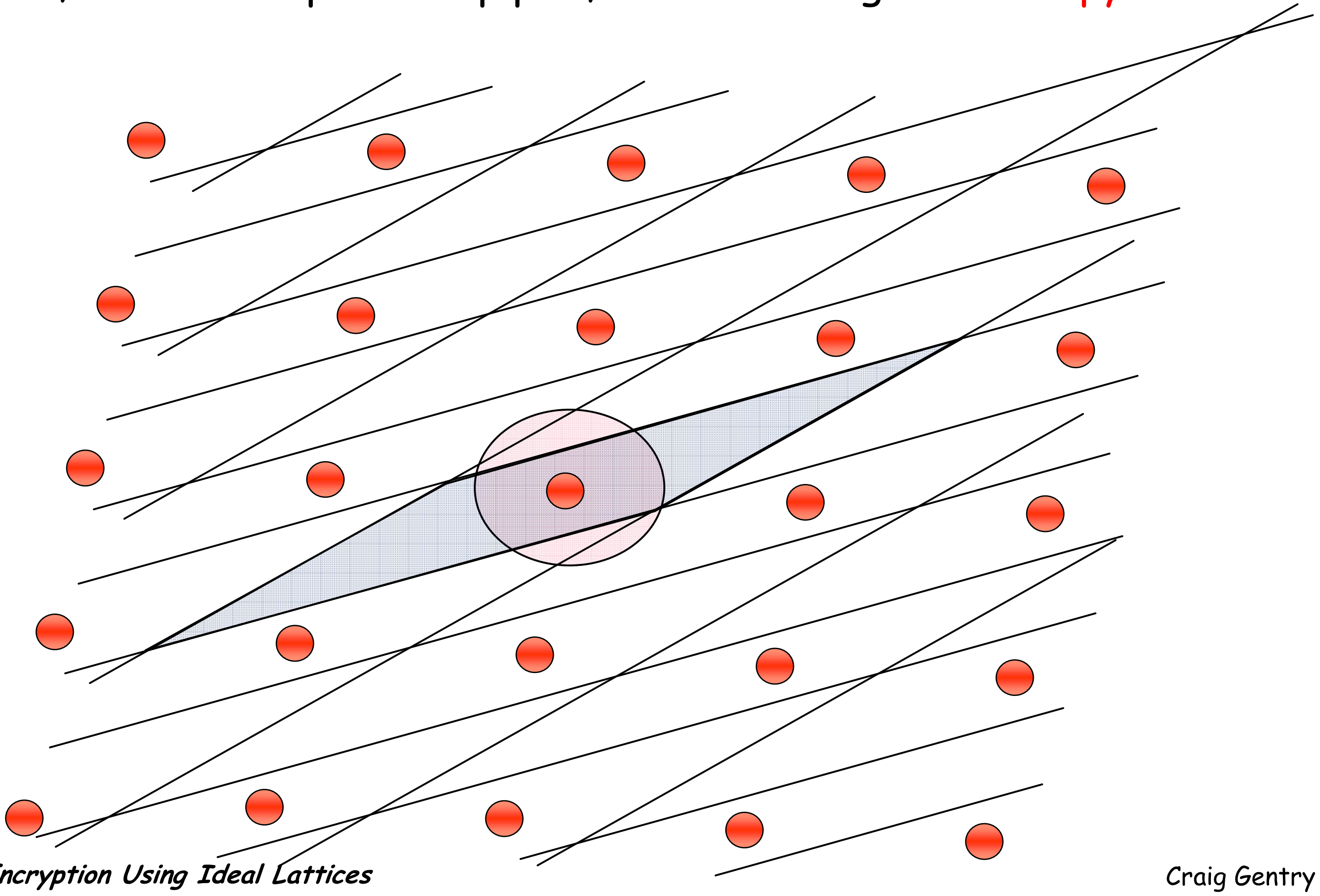
- If b = 0, t is uniform in blue parallelepiped.
- If b = 1, t is in blue parallelepiped, but according to a clumpy distribution.



Security of the Initial Ideal Lattice Scheme

Distributional CVP: Generate basis B_{pk} for ideal lattice J using KeyGen. Set bit b.

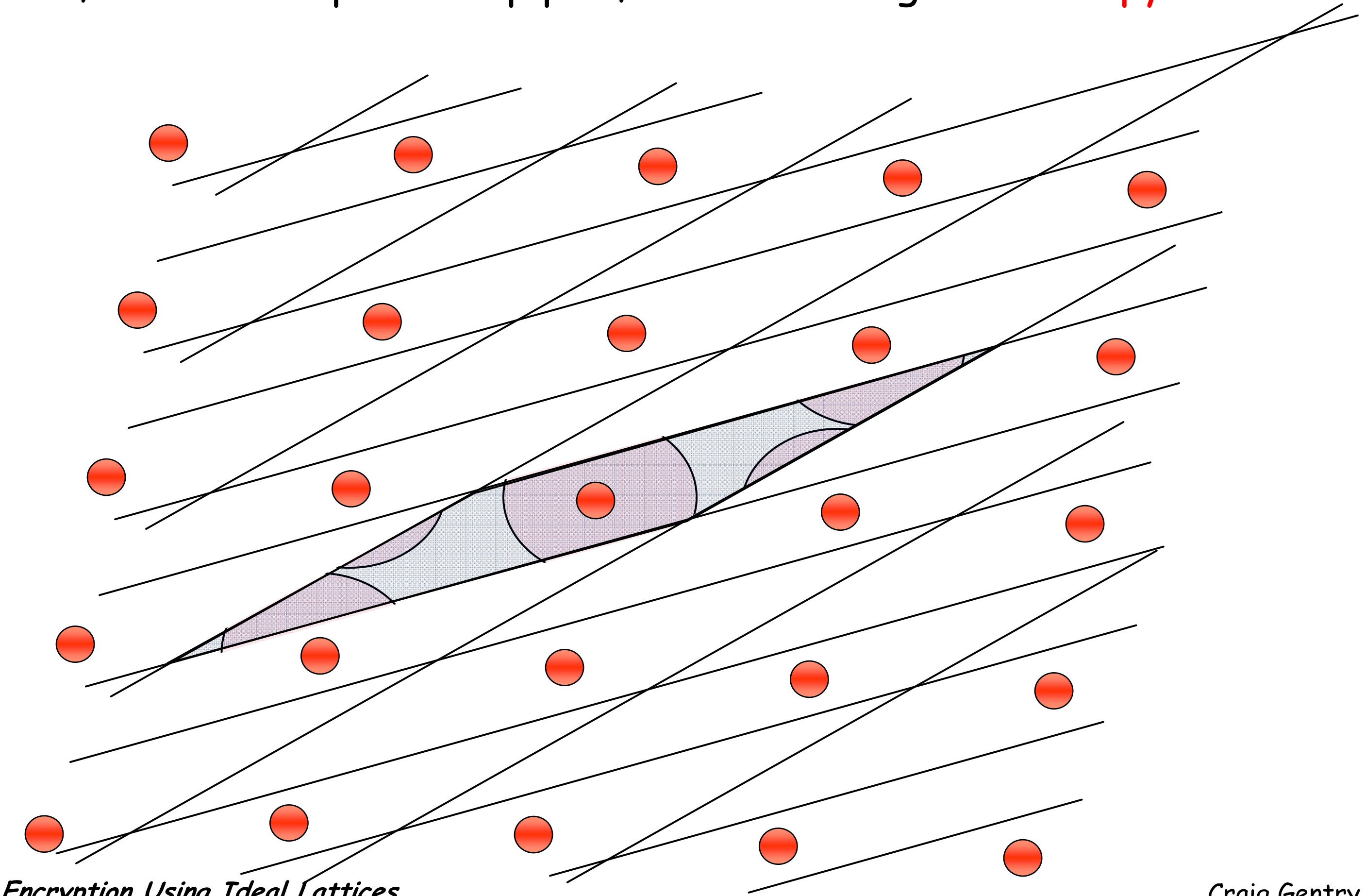
- If b = 0, t is uniform in blue parallelepiped.
- If b = 1, t is in blue parallelepiped, but according to a clumpy distribution.



Security of the Initial Ideal Lattice Scheme

Distributional CVP: Generate basis B_{pk} for ideal lattice J using KeyGen. Set bit b.

- If b = 0, t is uniform in blue parallelepiped.
- If b = 1, t is in blue parallelepiped, but according to a clumpy distribution.



Security

- Distributional CVP: Generate basis B_{pk} for ideal lattice J using KeyGen. Set bit b.
    - If b = 0, t is uniform in blue parallelepiped.
    - If b = 1, t is in blue parallelepiped, but according to a clumpy distribution (say, of radius r).
- Security proof sketch:
    - If b=1, t can be used to validly encrypt m, as follows:
        - Let s be a short vector in I, such that the ideal (s) is relatively prime to the ideal J.
        - Output $c \leftarrow m + s \times t \bmod B_{pk}$.
    - If b=0, then $c \leftarrow m + s \times t \bmod B_{pk}$ will be random modulo J and independent of m.

Circuit Privacy

- Algorithm “Randomize”:
  - Applied to outputs of Encrypt or Evaluate, it induces statistically equivalent distributions.
  - The Idea: Add a random encryption of 0 whose “error space” is huge in comparison to the “error space” ciphertexts output by Encrypt or Evaluate.
  - New error space for Evaluate is $B(r_{Dec}/m)$ for super-polynomial m, but no problem...

Let Us Revisit the Initial Construction to Get a Better Security Result…

- Parameters: Ring $R = Z[x]/(f(x))$, basis B_I of "small" ideal lattice I. Radii R_{Dec} and R_{Enc} as before. The operations "+" and "×" are in R.
- KeyGen: Output "good" and "bad" bases (B_{sk}, B_{pk}) of a "big" ideal lattice J, which is relatively prime to I – i.e., $I + J = R$. Plaintext space: the cosets of I.
- Encrypt(B_{pk}, m): Set $m' \xleftarrow{R} (m+I) \cap B(r_{Enc})$. Set $c \leftarrow m' \bmod B_{pk}$.
- Decrypt(B_{sk}, c): Output $(c \bmod B_{sk}) \bmod B_I \rightarrow m$
- Add(B_{pk}, c_1, c_2): Output $c \leftarrow c_1 + c_2 \bmod B_{pk}$
- Mult(B_{pk}, c_1, c_2): Output $c \leftarrow c_1 \times c_2 \bmod B_{pk}$, which is in $m_1' \times m_2' + J$

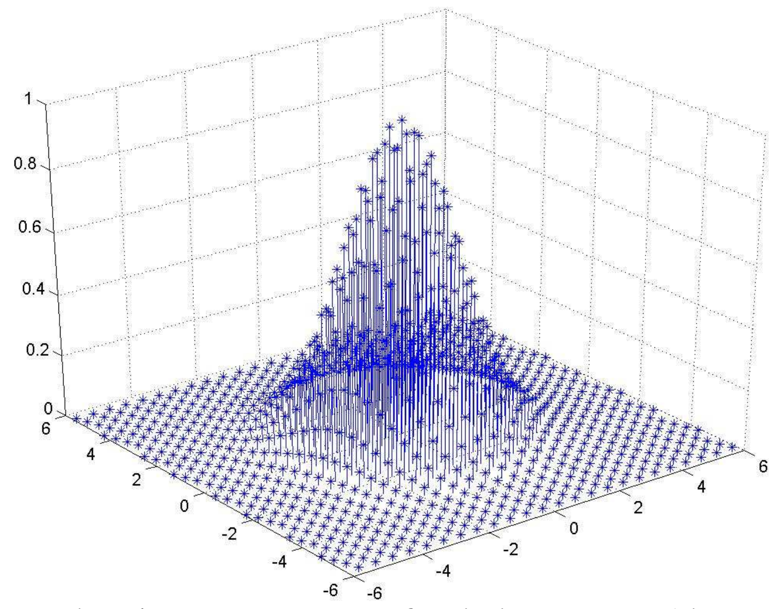
Let Us Revisit the Initial Construction to Get a Better Security Result…

- Parameters: Ring $R = Z[x]/(f(x))$, basis B_I of "small" ideal lattice I. Radii R_{Dec} and R_{Enc} as before. The operations "+" and "×" are in R.
- KeyGen: Output "good" and "bad" bases (B_{sk}, B_{pk}) of a "big" ideal lattice J, which is relatively prime to I – i.e., $I + J = R$. Plaintext space: the cosets of I.
- Encrypt(B_{pk}, m): Set $m' \xleftarrow{R} (m+I) \cap B(r_{Enc})$. Set $c \leftarrow m' \bmod B_{pk}$.
- Decrypt(B_{sk}, c): Output $(c \bmod B_{sk}) \bmod B_I \rightarrow m$
- Add(B_{pk}, c_1, c_2): Output $c \leftarrow c_1 + c_2 \bmod B_{pk}$
- Mult(B_{pk}, c_1, c_2): Output $c \leftarrow c_1 \times c_2 \bmod B_{pk}$, which is in $m_1' \times m_2' + J$

First step: Sample from m+I according to a Gaussian distribution.

Discrete Gaussian Distributions

- We modify our initial construction to use discrete Gaussian distributions over lattices.
- Sum of 2 discrete Gaussian distribution is statistically equivalent to another discrete Gaussian distribution.



Used without permission of Oded Regev. He'd probably let me if I asked though. Thanks Oded!

Security Inner Ideal Membership Problem (IIMP)

- The IIMP: Fix R, B_I, and real m_{IIMP}. Run $(B_{sk}, B_{pk}) \leftarrow$ KeyGen(R, B_I), bases for some ideal J. Set $b \xleftarrow{R} \{0,1\}$.
  - If b=0, one samples $v \leftarrow$ Gauss(I, s, 0) and sets $t \leftarrow v \bmod B_{pk}$.
  - If b=1, one samples $v \leftarrow$ Gauss(Z^n, s, 0) and sets $t \leftarrow v \bmod B_{pk}$.
  - Given (B_{pk}, t) and the fixed values, decide b.
- Security proof sketch:
  - Set $w \leftarrow$ Gauss(I, s, $-m_b$). Set $c \leftarrow m_b+w+v \bmod B_{pk}$.
  - If b=0, $(c \bmod B_{sk}) \bmod B_I = (m_b+w+v) \bmod B_I = m_b$.
  - If b=1, $(c \bmod B_{sk}) \bmod B_I = (m_b+w+v) \bmod B_I$ = random.

From Modified IIMP

- The MIIMP: Like the IIMP, except $m_{MIIMP} < m_{IIMP} \cdot \varepsilon / (n \cdot |B_I|)$ and
  - If b=0, one sets $v \leftarrow I$ so that $|v| < m_{MIIMP}$
  - If b=1, one sets v not in I so that $|v| < m_{MIIMP}$
  - Given (B_{pk}, $t = v \bmod B_{pk}$) and the fixed values, decide b.
- Sketch of reduction to IIMP:
  - Set u to be very short, but random modulo I.
  - Set $t' \leftarrow u \times t + \text{Gauss}(I, m_{IIMP}, 0) \bmod B_{pk}$.
  - IIMP instance is (B_{pk}, t').
  - If b = 0, then indeed t' is "in the inner ideal."
  - If b = 1, t' is uniformly random wrt I.

From Average-Case CVP Using Hensel Lifting

- Average-case CVP: Set $m_{ACVP} < m_{MIIMP}/(\gamma_{MULT}(R) \cdot \sqrt{n})$. Set v such that $|v| < m_{ACVP}$, and set $t \leftarrow v \bmod B_{pk}$.
  - Given (B_{pk}, t), output v. (This is a search problem!)
- Sketch of reduction to MIIMP:
  - Use MIIMP-oracle to get $v_1 \leftarrow v \bmod B_I$.
  - Set w to be a short vector in I^{-1}, and use the MIIMP-oracle to get $v_2' \leftarrow w \times (v-v_1) \bmod B_I$. This gives $v_2 \leftarrow v \bmod I^2$.
  - Etc.
  - Given $v_k = v \bmod I^k$, we know $v_k - v$ is in I^k. For large enough k, we can use LLL to solve this CVP in poly time (to get v).

Average-Case / Worst-Case Connection for Ideal Lattices?

- Yes
- First ac / wc connection where ac problem is for ideal lattices.
- First ac / wc connection where ac lattice has same dimension as wc lattice (usually the ac lattice is larger).
- I need quantum computation for the reduction…

What is the average-case distribution?

- What is a random ideal?
- Our definition: uniformly random among ideals whose norm (i.e., determinant) is in a fixed interval – e.g., $[n^{cn}, 2n^{cn}]$.

How to Generate (a Basis of) a Random Ideal…

- Our Technique: Adapt Kalai's technique for generating a random factored number.
- We generate a random factored *norm* N of an ideal in R.
- It is easy to generate bases for an ideal whose norm is prime.
- We multiply together the bases of the individual primes to get a basis whose norm is N.

KeyGen

- Goal: Ideal J, together with a good independent set for J^{-1}.
- Generate a random ideal K with norm in $[n^{cn}, 2n^{cn}]$.
- Generate $v \leftarrow \text{Gauss}(K^{-1}, s, t\cdot\mathbf{e}_1)$. I.e., v almost equals $t \cdot \mathbf{e}_1$.
- Set $J \leftarrow K \cdot (v)$.
- Already have a somewhat good independent set for K – i.e., $\{\mathbf{e}_i\}$.
- Our good independent set for J^{-1} is $\{\mathbf{e}_i/v\}$.
- Proving that J has a nice average-case distribution (in a different interval) uses properties of discrete Gaussian distributions.

How Do We "Randomize" a Worst-Case Ideal?

- Given worst-case CVP instance (B_M, u), how do we randomize it to obtain average-case instance (B_J, t), such that solving the ac instance helps us solve the wc instance?
- First, we multiply M by a random ideal K. Intuitively, the *shape* of MK is essentially independent of M.
- Next, we multiply by $v \leftarrow \text{Gauss}((MK)^{-1}, s, t \cdot \mathbf{e}_1)$ to "divide out" the *algebraic* dependence on M.
- We set $J \leftarrow MK \cdot (v)$ and $t \leftarrow u \times w_K \times v$, where w_k is a very short vector in K (of length poly(n)).
- But, wait, our method of generating a random K didn't also give a short w_K in K…

How to Generate a Random Ideal with a Short Vector in It… Quantumly

- Generate the short w first via w $\leftarrow$ Gauss(Z^n, s, $t \cdot \mathbf{e}_1$)
- *Factor* the ideal (w) by factoring the norm of (w) using Shor's quantum factoring algorithm.
- Set K to be a random divisor of (w).

Worst-Case CVP to Independent Vector Improvement Problem (IVIP)

- [Regev]: uses quantum computation
- Superposition 1: Gaussian distribution (Z^n, s, 0).
- Superposition 2: Reduce each point in the above distribution modulo a basis B_L for the lattice L.
  - If there is a classical CVP oracle for L that solves it when t is within $s\sqrt{n}$ of a lattice point, this reduction is *reversible*.
- Superposition 3: Fourier transform to get distribution (L*, 1/s, 0).
- Measure, to get a point in L* of length at most $\sqrt{n}/s$.

IVIP to Shortest Independent Vector Problem

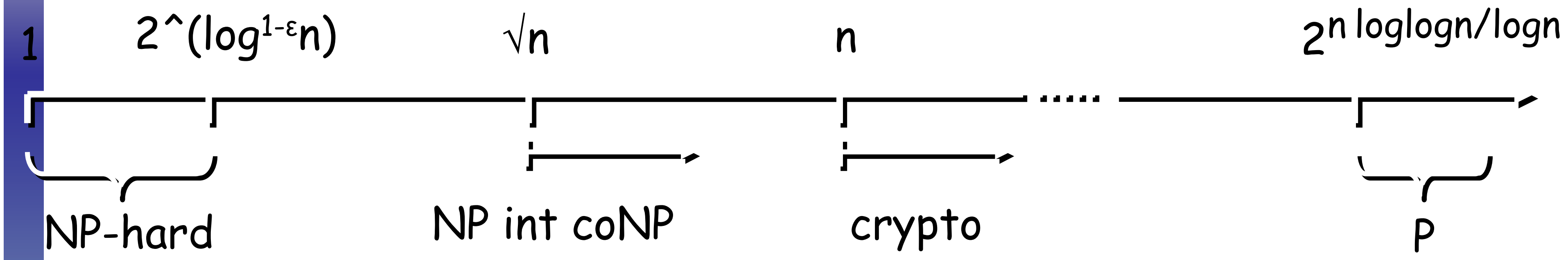
- The SIVP: Generate n linearly independent vectors in a given lattice L, all of length at most $m_{SIVP} \cdot \lambda_n(L)$.
- Sketch of reduction to IVIP
  - Given M_0, use the IVIP oracle to find an independent set of M_0^{-1} with vectors of length at most $1/m_{IVIP}$.
  - Set $v \leftarrow \text{Gauss}(M_0^{-1}, s/m_{IVIP}, (t/m_{IVIP})\cdot e_1)$ and $M_1 \leftarrow M_0\cdot(v)$.
  - Recurse.
- Result: Let $d_{SIVP} = 3^{1/n}\cdot d_{IVIP}$. If there is an algorithm that solves IVIP for $m_{IVIP} = 8 \cdot \lambda_{MULT}(R) \cdot n^{2.5} \cdot \log n$ whenever the given ideal has $\det(M)^{1/n} > d_{IVIP}$, then there is an algorithm that solves SIVP for approximation factor d_{SIVP}.

Correctness

Correctness: Decryption works on Evaluate($B_{J,pk}$, C, ψ_1, … ψ_t) if $C(\pi_1+i_1, \dots, \pi_t+i_t)$ is the disting. rep. of its coset w.r.t. $B_{J,sk}$.

- Ciphertext $\psi_k = \pi_k + i_k + j_k$, with i in I and j in J.
- Evaluate($B_{J,pk}$, C, $\psi_1, \dots, \psi_t$) = $C(\pi_1+i_1+j_1, \dots, \pi_t+i_t+j_t)$
- in $C(\pi_1+i_1, \dots, \pi_t+i_t)$
- If $C(\pi_1+i_1, \dots, \pi_t+i_t)$ is the disting. rep. of its coset of J w.r.t. $B_{J,sk}$, which is true if $C(Y, \dots, Y)$ is a subset of R mod $B_{J,sk}$, then Decrypt returns $C(\pi_1+i_1, \dots, \pi_t+i_t) \bmod B_I = C(\pi_1, \dots, \pi_t) \bmod B_I$.

Cryptographically Hard Problems Over Lattices

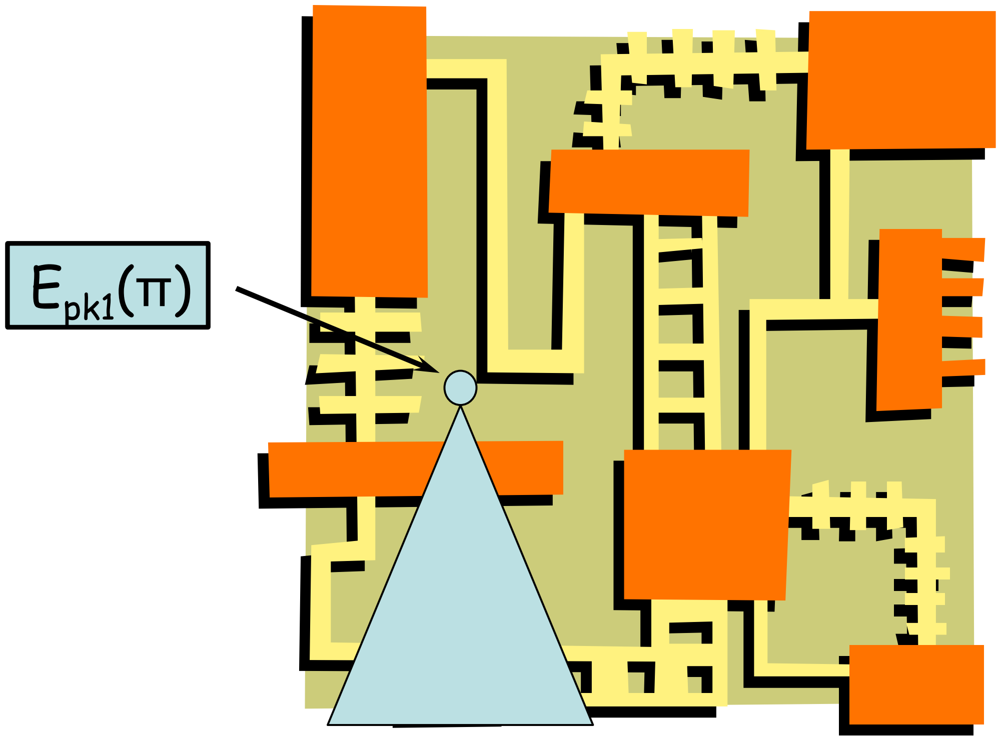


- The LLL algorithm (with Babai's modifications) can approximate CVP to within a factor of about 2^n in polynomial time.
- We do not know how to do better in general.

Let us review our additively homomorphic scheme…

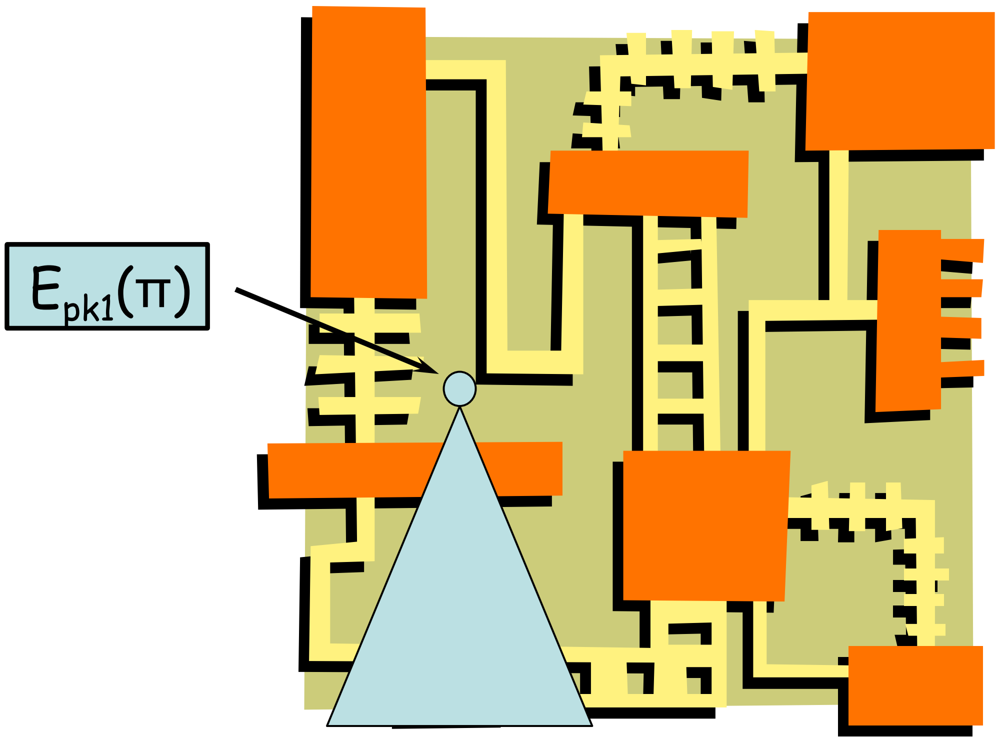
- Global Parameters: r_{Dec}, r_{Enc}, Z^n, and a basis B_H of an additive subgroup H of Z^n. E.g., H could be the vectors with even coefficient sum. Plaintext space is the set of "distinguished reps" of the cosets of H.
- KeyGen: Secret and public bases B_{sk} and B_{pk} of some lattice L, where B_{sk} circumscribes a ball of radius r_{Dec}.
- Encrypt(B_{pk}, m): Set $m' \leftarrow^R (m+H) \cap B(r_{Enc})$. Set $c \leftarrow m' \bmod B_{pk}$.
- Decrypt(B_{sk}, c): Set $m \leftarrow (c \bmod B_{sk}) \bmod B_H$. Note: $m' = (c \bmod B_{sk})$.
- Add(B_{PK}, c_1, c_2): Set $c \leftarrow c_1 + c_2 \bmod B_{PK}$, which is in $m'_1 + m'_2 + L$.
- Correctness: Let C be a mod-B_H circuit that adds at most r_{Dec}/r_{Enc} plaintexts. Then, Evaluate(B_{pk}, C, c_1, ..., c_t) decrypts correctly since:
  1) $m'_1+\ldots+m'_t = c_1+\ldots+c_t \bmod B_{sk}$, since it is in the secret parallelepiped.
  2) $m_1+\ldots+m_t = m'_1+\ldots+m'_t \bmod B_H$.

How Does It All Work Together?



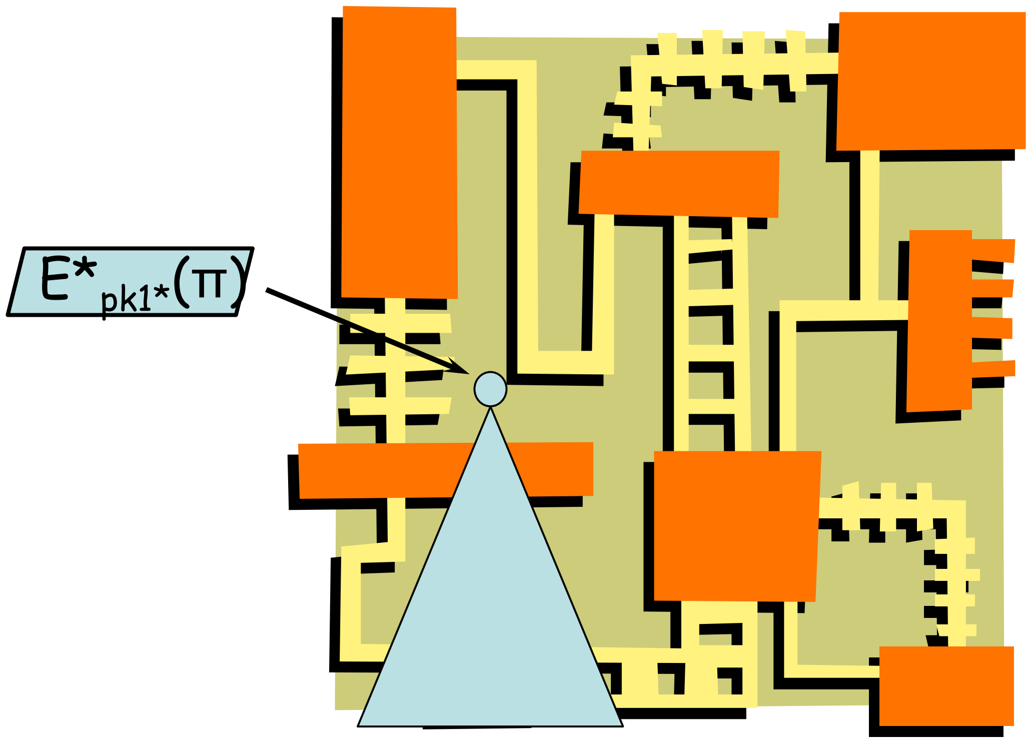
How Does It All Work Together?

E is the initial scheme.
E* has the squashed dec circuit.



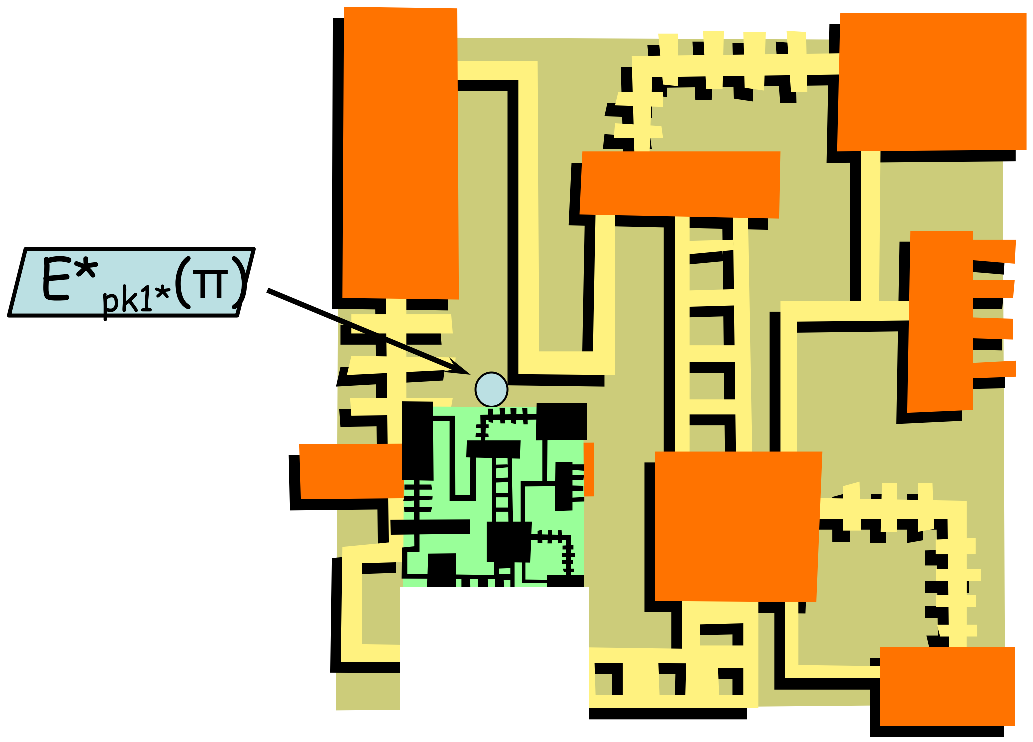
How Does It All Work Together?

E is the initial scheme.
E* has the squashed dec circuit.



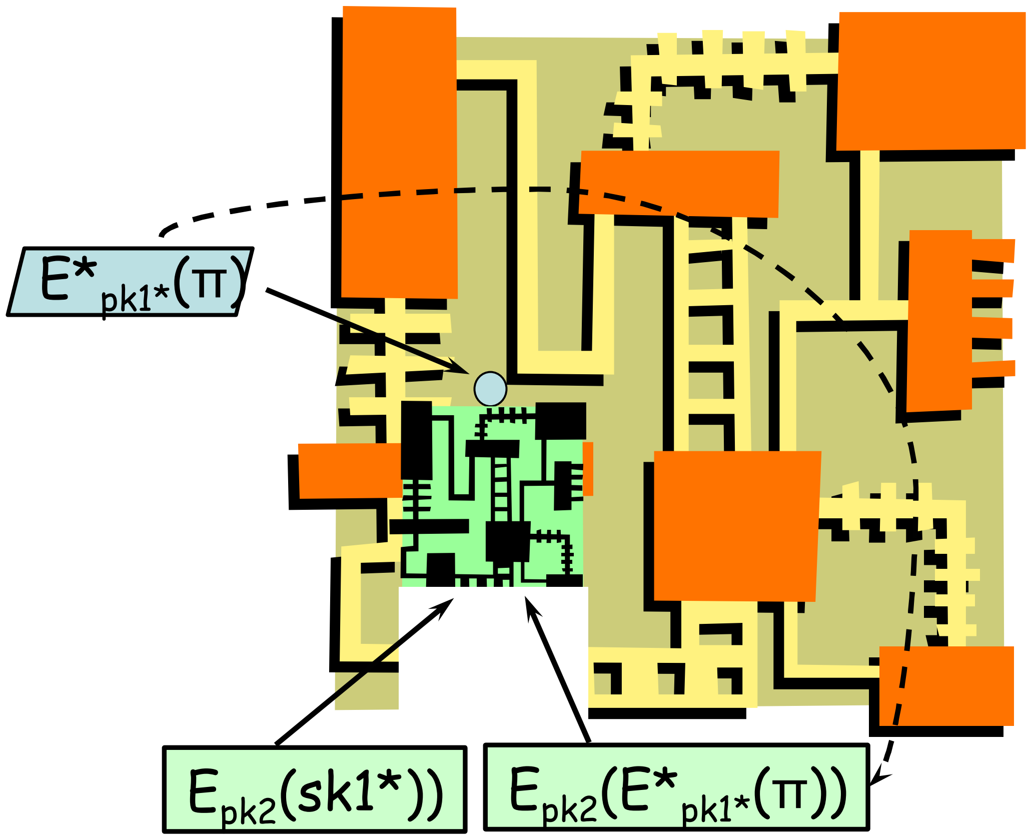
How Does It All Work Together?

E is the initial scheme.
E* has the squashed dec circuit.



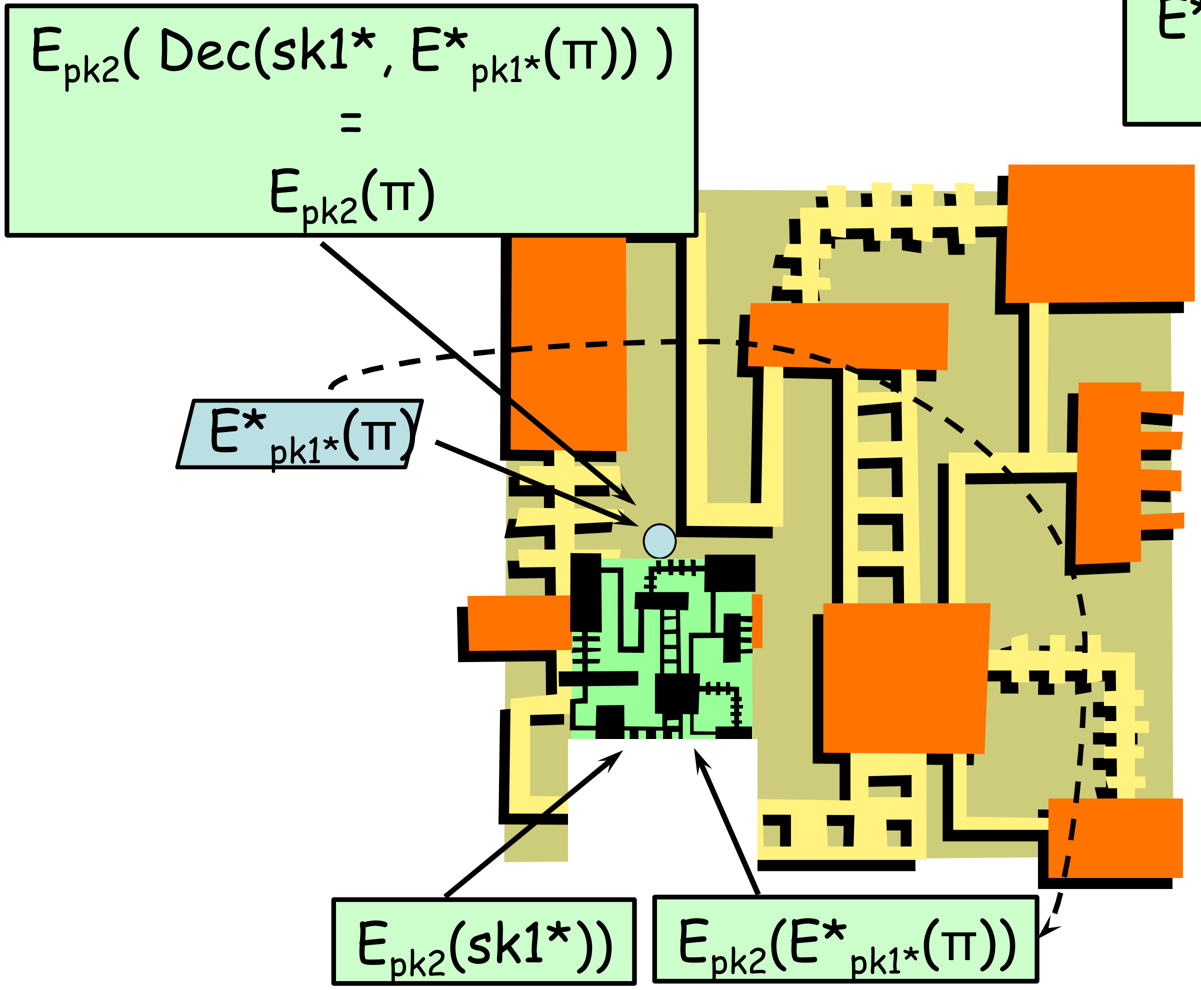
How Does It All Work Together?

E is the initial scheme. E* has the squashed dec circuit.



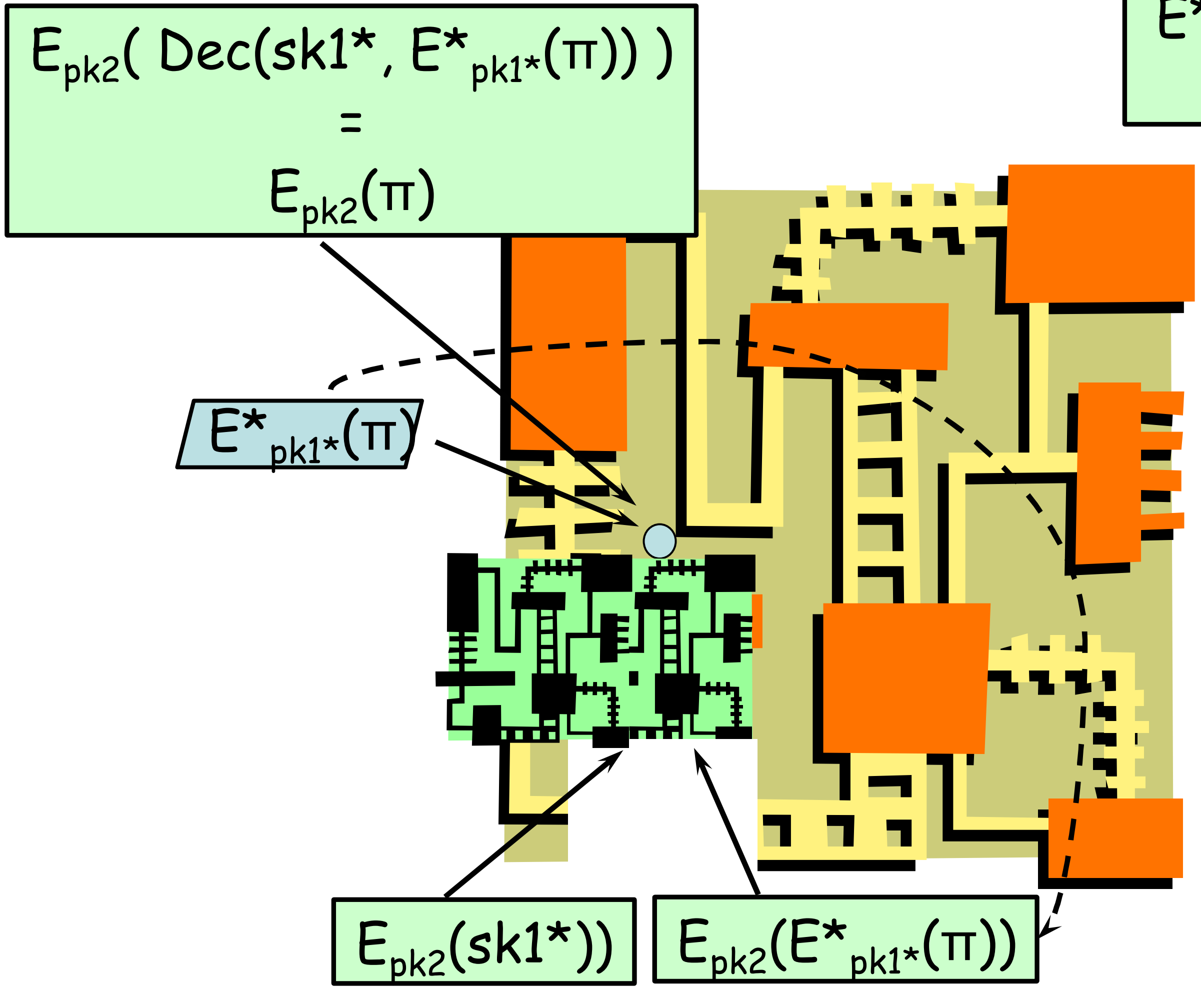
How Does It All Work Together?

E is the initial scheme.
E* has the squashed dec circuit.


$E_{pk2}(\text{ Dec}(sk1^*, E^*_{pk1^*}(\pi)) \text{ })$
$=$
$E_{pk2}(\pi)$
$E^*_{pk1^*}(\pi)$
$E_{pk2}(sk1^*))$
$E_{pk2}(E^*_{pk1^*}(\pi))$

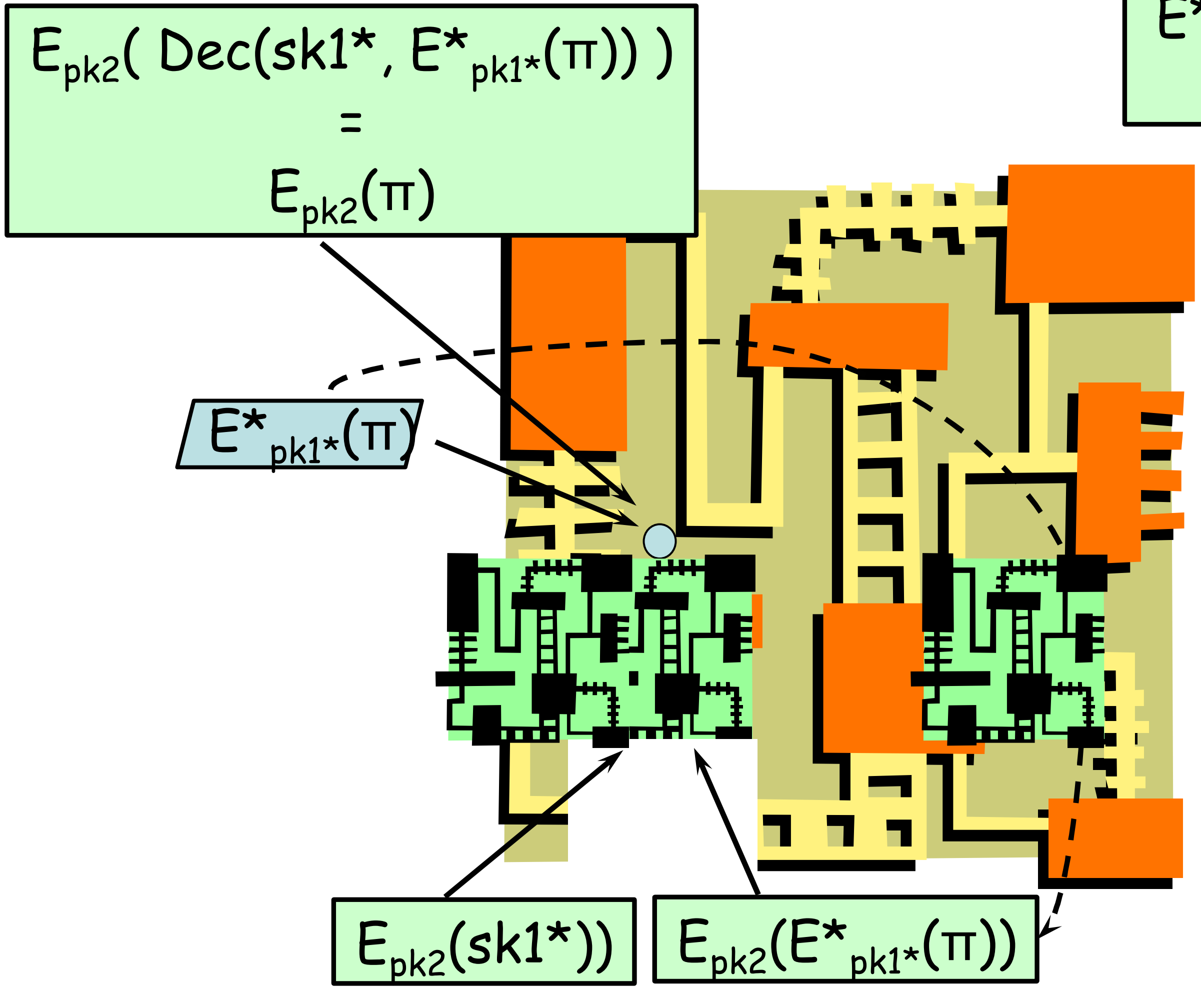
How Does It All Work Together?

E is the initial scheme.
E* has the squashed dec circuit.


$E_{pk2}(\text{Dec}(sk1^*, E^*_{pk1^*}(\pi)))$
$=$
$E_{pk2}(\pi)$
$E^*_{pk1^*}(\pi)$
$E_{pk2}(sk1^*)$
$E_{pk2}(E^*_{pk1^*}(\pi))$

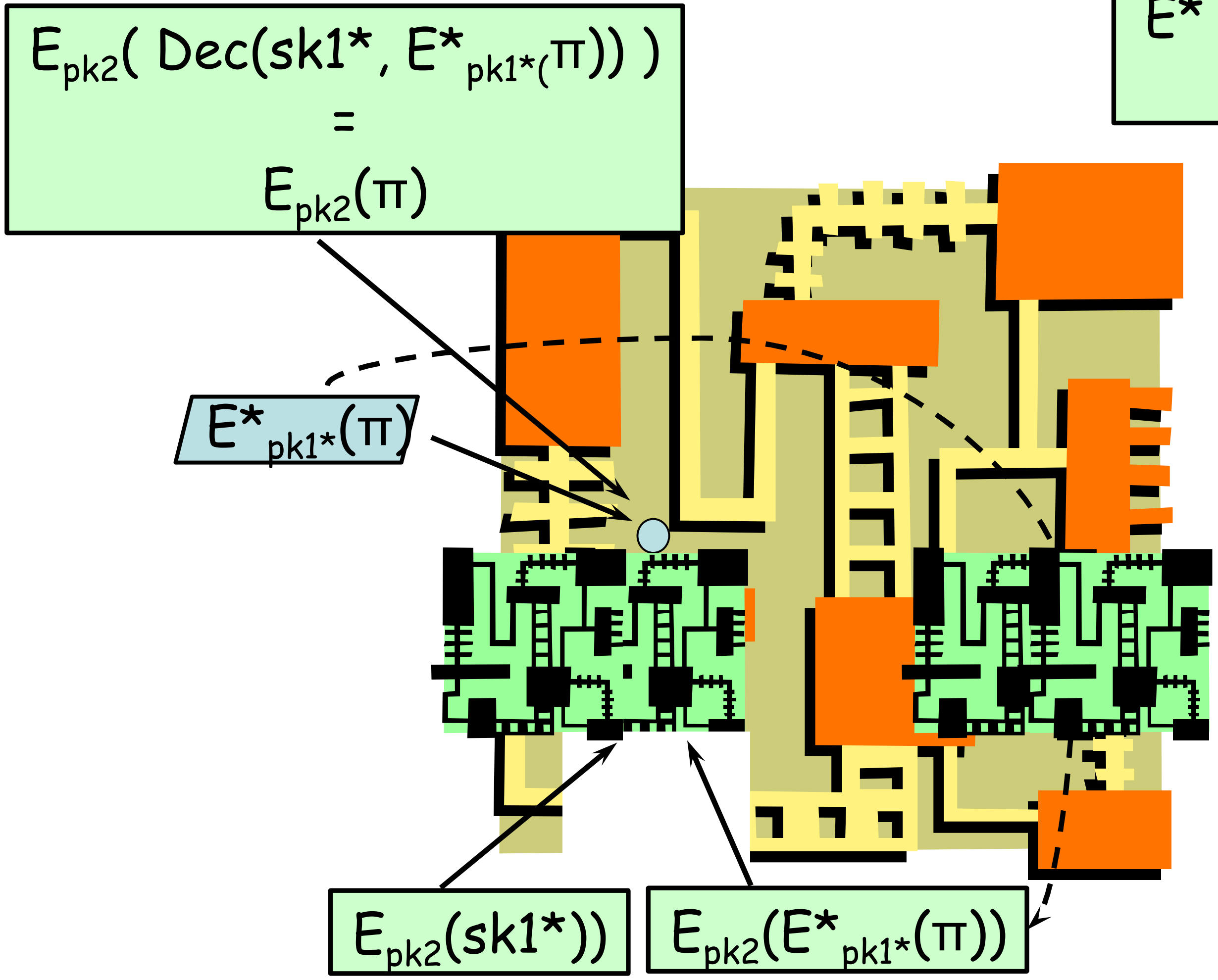
How Does It All Work Together?

E is the initial scheme.
E* has the squashed dec circuit.


$E_{pk2}(\ Dec(sk1^*, E^*_{pk1^*}(\pi))\ )$
$=$
$E_{pk2}(\pi)$
$E^*_{pk1^*}(\pi)$
$E_{pk2}(sk1^*))$
$E_{pk2}(E^*_{pk1^*}(\pi))$

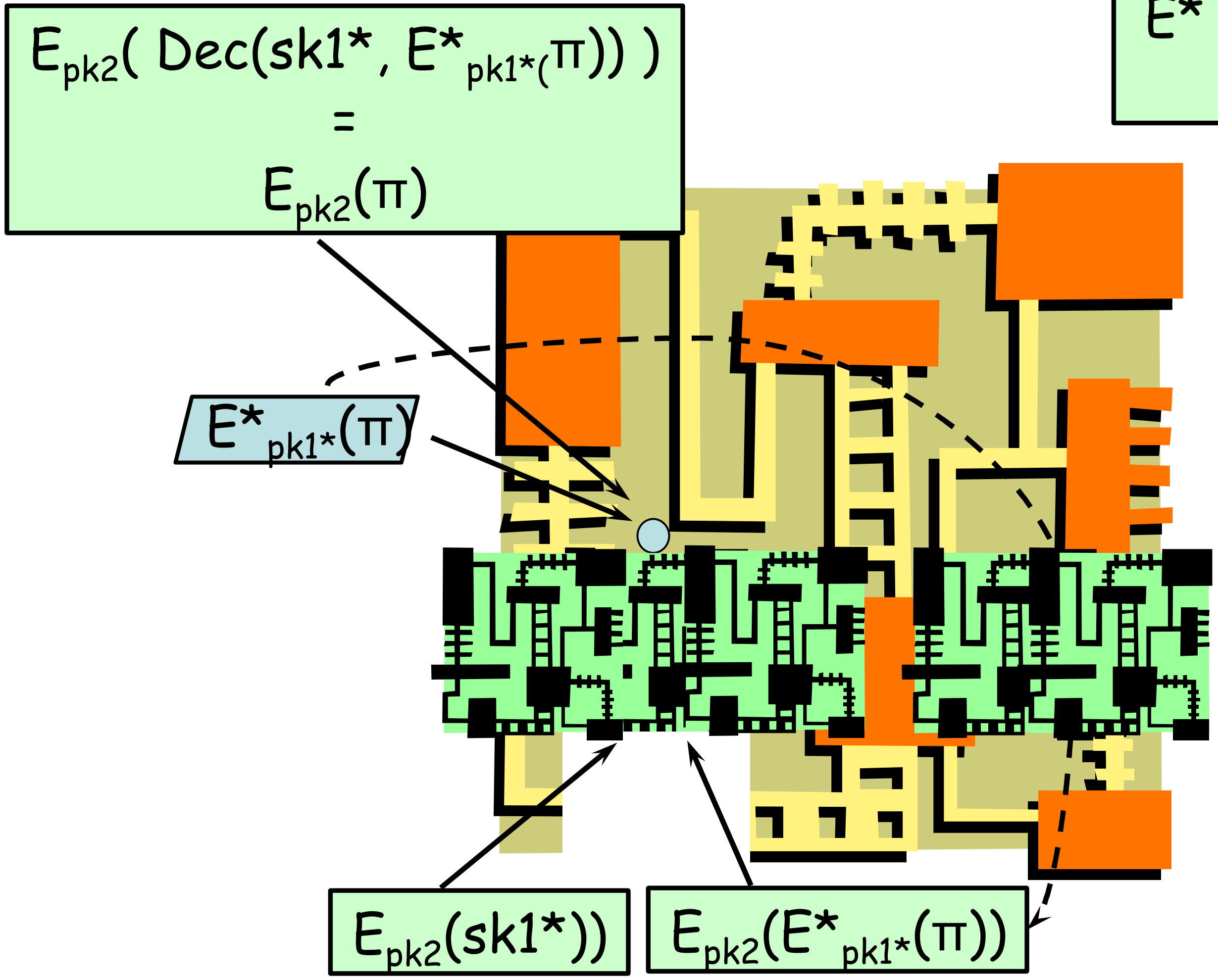
How Does It All Work Together?

E is the initial scheme.
E* has the squashed dec circuit.


$E_{pk2}(\text{ Dec}(sk1^*, E^*_{pk1^*}(\pi)) \text{)}$
=
$E_{pk2}(\pi)$
$E^*_{pk1^*}(\pi)$
$E_{pk2}(sk1^*))$
$E_{pk2}(E^*_{pk1^*}(\pi))$

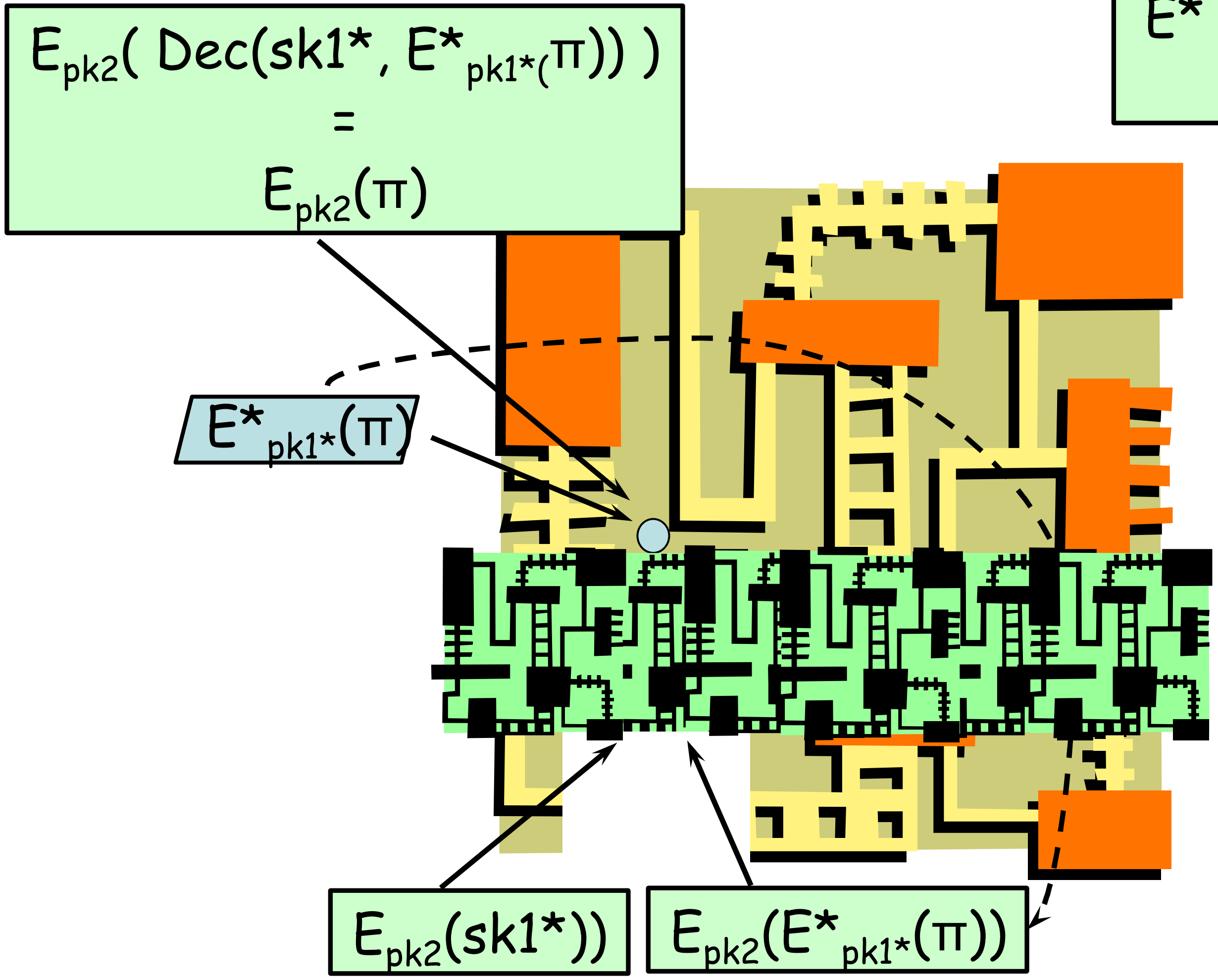
How Does It All Work Together?

E is the initial scheme.
E* has the squashed dec circuit.


$E_{pk2}(\text{ Dec}(sk1^*, E^*_{pk1^*}(\pi)) \text{ })$
=
$E_{pk2}(\pi)$
$E^*_{pk1^*}(\pi)$
$E_{pk2}(sk1^*))$
$E_{pk2}(E^*_{pk1^*}(\pi))$

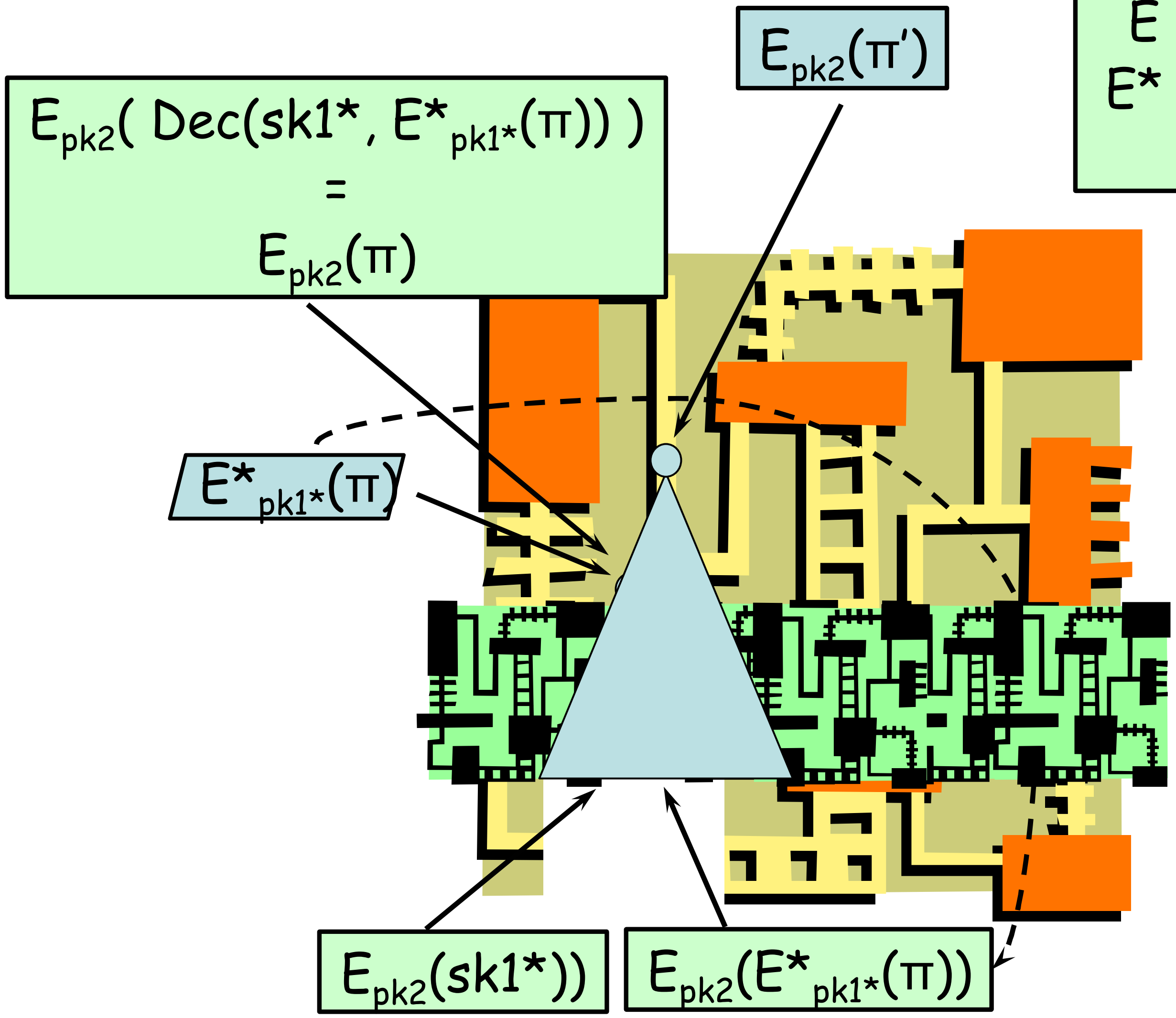
How Does It All Work Together?

E is the initial scheme.
E* has the squashed dec circuit.


$E_{pk2}(\ Dec(sk1^*, E^*_{pk1^*}(\pi))\ )$
$=$
$E_{pk2}(\pi)$
$E^*_{pk1^*}(\pi)$
$E_{pk2}(sk1^*))$
$E_{pk2}(E^*_{pk1^*}(\pi))$

How Does It All Work Together?

E is the initial scheme. E* has the squashed dec circuit.


$E_{pk2}(\pi')$
$E_{pk2}(\text{ Dec}(sk1^*, E^*_{pk1^*}(\pi)) \text{ })$
$=$
$E_{pk2}(\pi)$
$E^*_{pk1^*}(\pi)$
$E_{pk2}(sk1^*))$
$E_{pk2}(E^*_{pk1^*}(\pi))$

And so on…