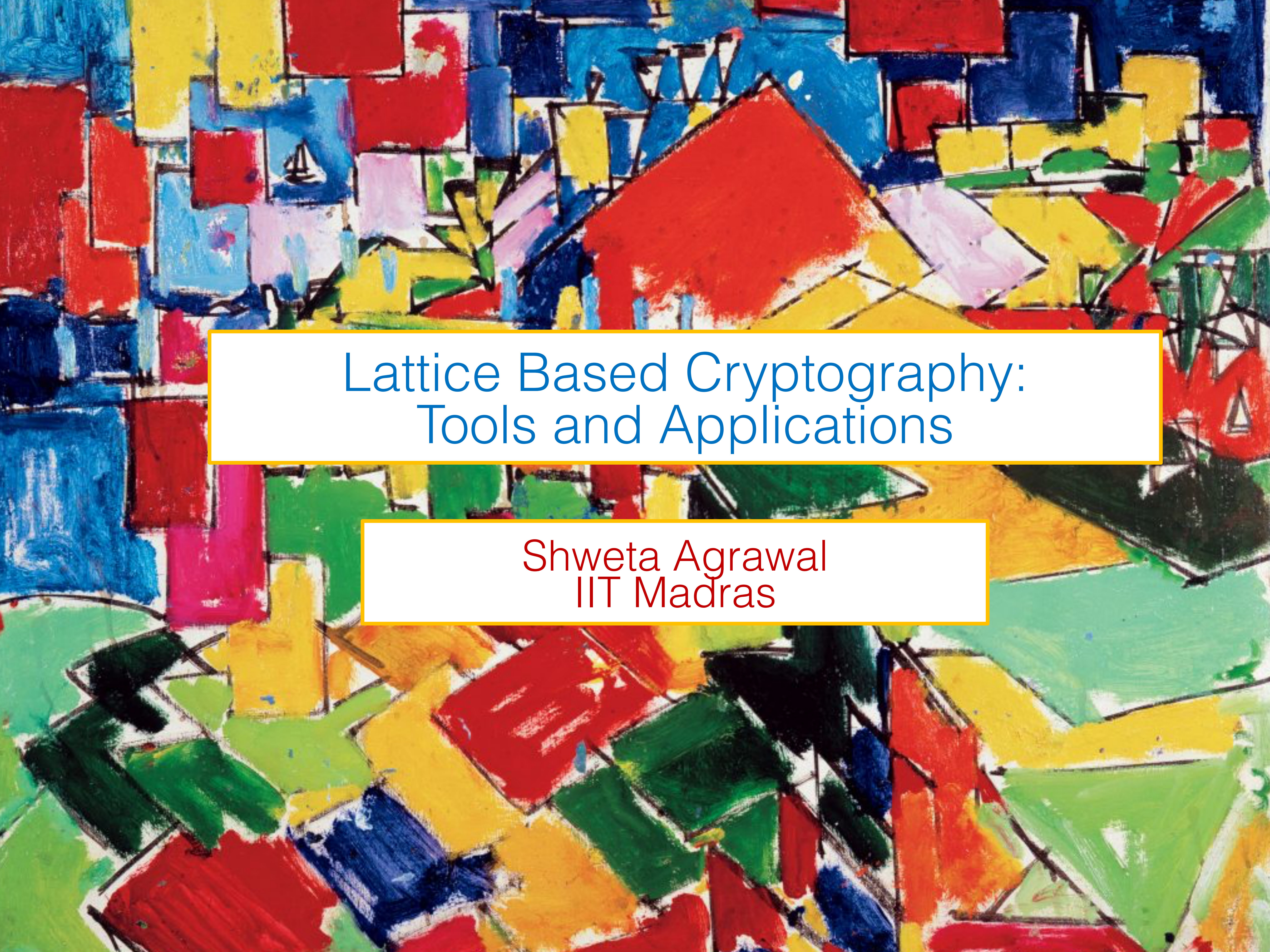


Lattice Based Cryptography: Tools and Applications

Shweta Agrawal
IIT Madras

Computing on Encrypted Data
Personalised Medicine

“The dream for tomorrow’s medicine is to understand the links between DNA and disease — and to tailor therapies accordingly. But scientists have a problem: how to keep genetic data and medical records secure while still enabling the massive, cloud-based analyses needed to make meaningful associations.”

Check Hayden, E. (2015). *Nature*, *519*, 400-401.

“You don’t look anything like the long haired, skinny kid I married 25 years ago. I need a DNA sample to make sure it’s still you.”

Computing on Encrypted Data
Personalised Medicine

"The dream for tomorrow's medicine is to understand the links between DNA and disease — and to tailor therapies accordingly. But scientists have a problem: how to keep genetic data and medical records secure while still enabling the massive, cloud-based analyses needed to make meaningful associations."

"You don't look anything like the long haired, skinny kid I married 25 years ago. I need a DNA sample to make sure it's still you."

Check Hayden, E. (2015). *Nature*, *519*, 400-401.

Doesn't FHE solve exactly this?

Access Control on Encrypted Data

Prof. Bob wants to store encrypted file so that:

- Other Professors or admin assistants of CS group can open it
- Encrypt file for each of them?
- If someone quits or new person joins? Re-encrypt ?
- Organizational nightmare !

Access Control on Encrypted Data

Prof. Bob wants to store encrypted file so that:

Access Control on Encrypted Data

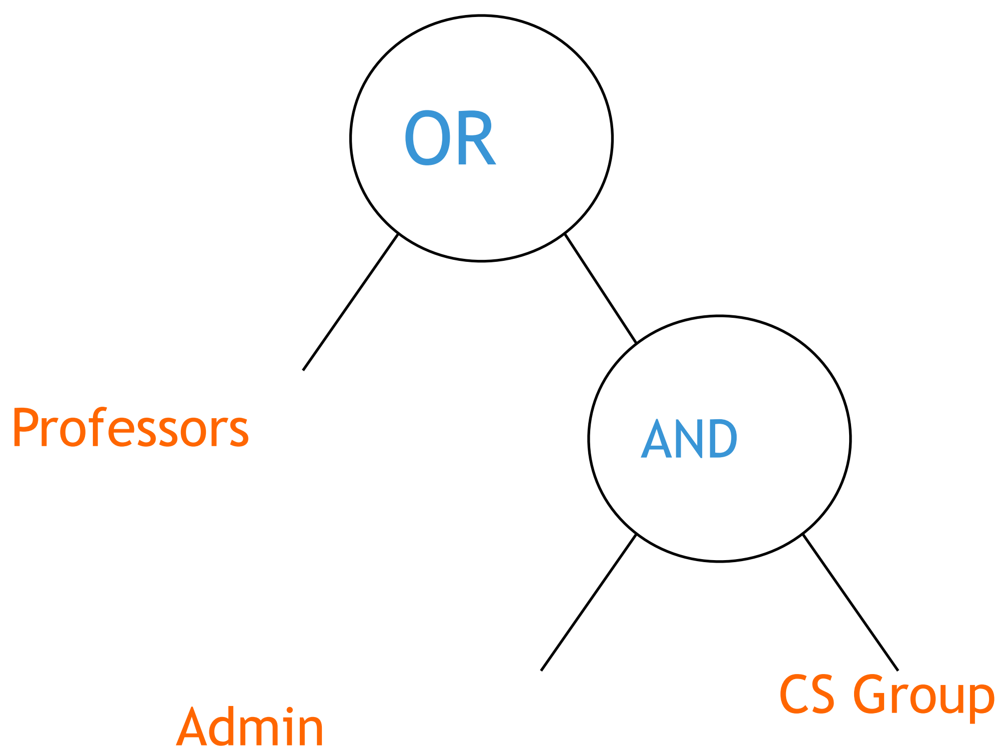
Prof. Bob wants to store encrypted file so that:

What he really wants:
Encryption for formula

Access Control on Encrypted Data

Prof. Bob wants to store encrypted file so that:

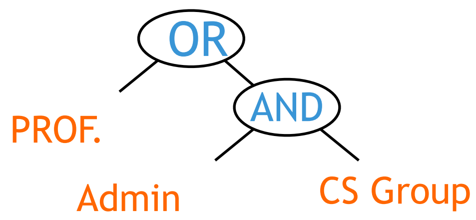
What he really wants:
Encryption for formula



What do we want?

What do we want?

What do we want?



What do we want?

What do we want?

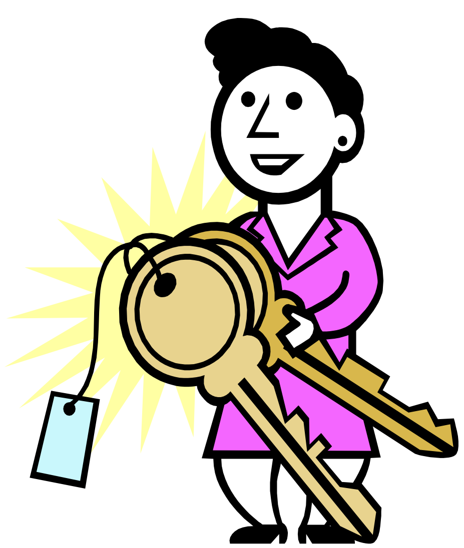
PROF OR {Admin AND CS}

What do we want?

PROF OR {Admin AND CS}

What do we want?

PROF OR {Admin AND CS}



Key Authority

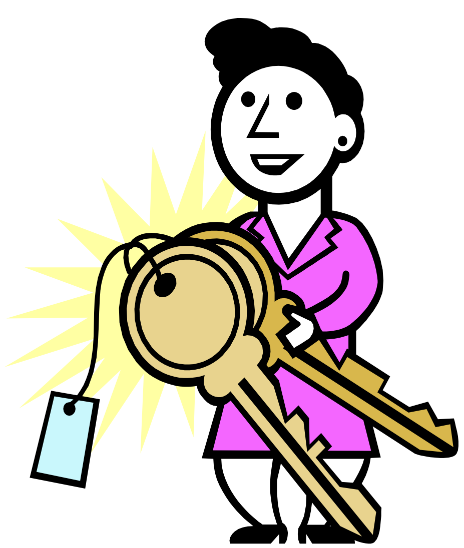
What do we want?

PROF OR {Admin AND CS}

Key Authority

What do we want?

PROF OR {Admin AND CS}

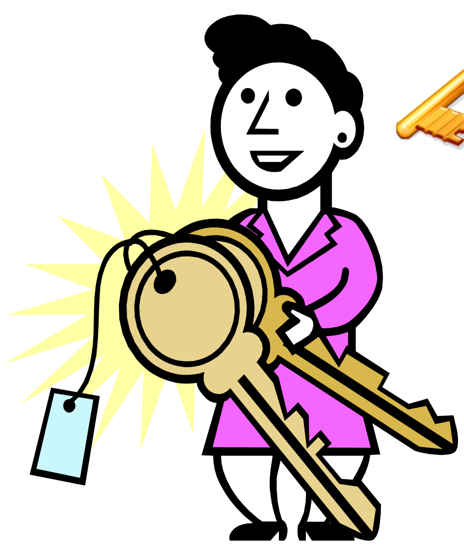


Key Authority

What do we want?

PROF OR {Admin AND CS}

PROF



Key Authority

What do we want?

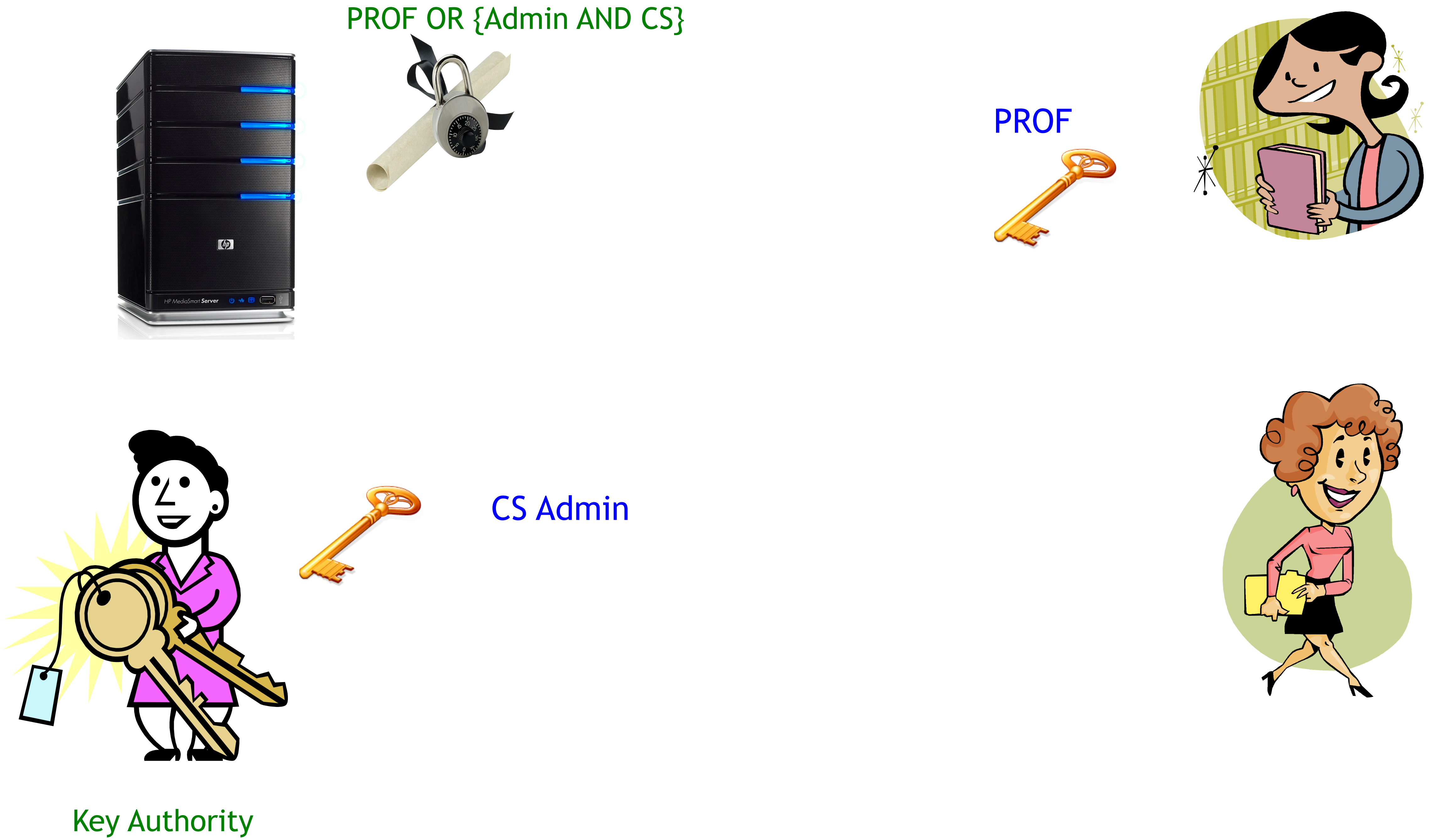
PROF OR {Admin AND CS}

PROF

CS Admin

Key Authority

What do we want?



What do we want?

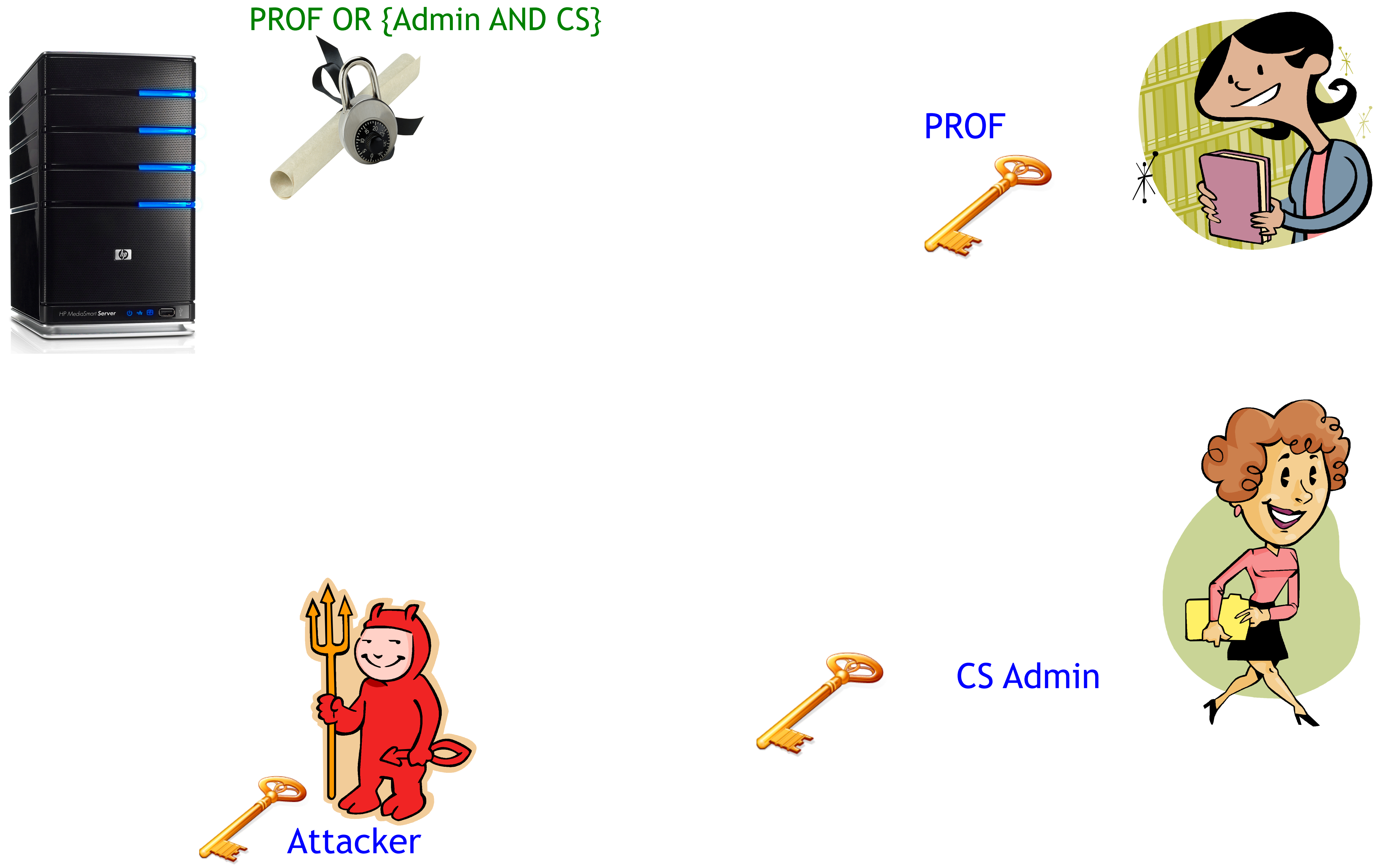
What do we want?

PROF OR {Admin AND CS}

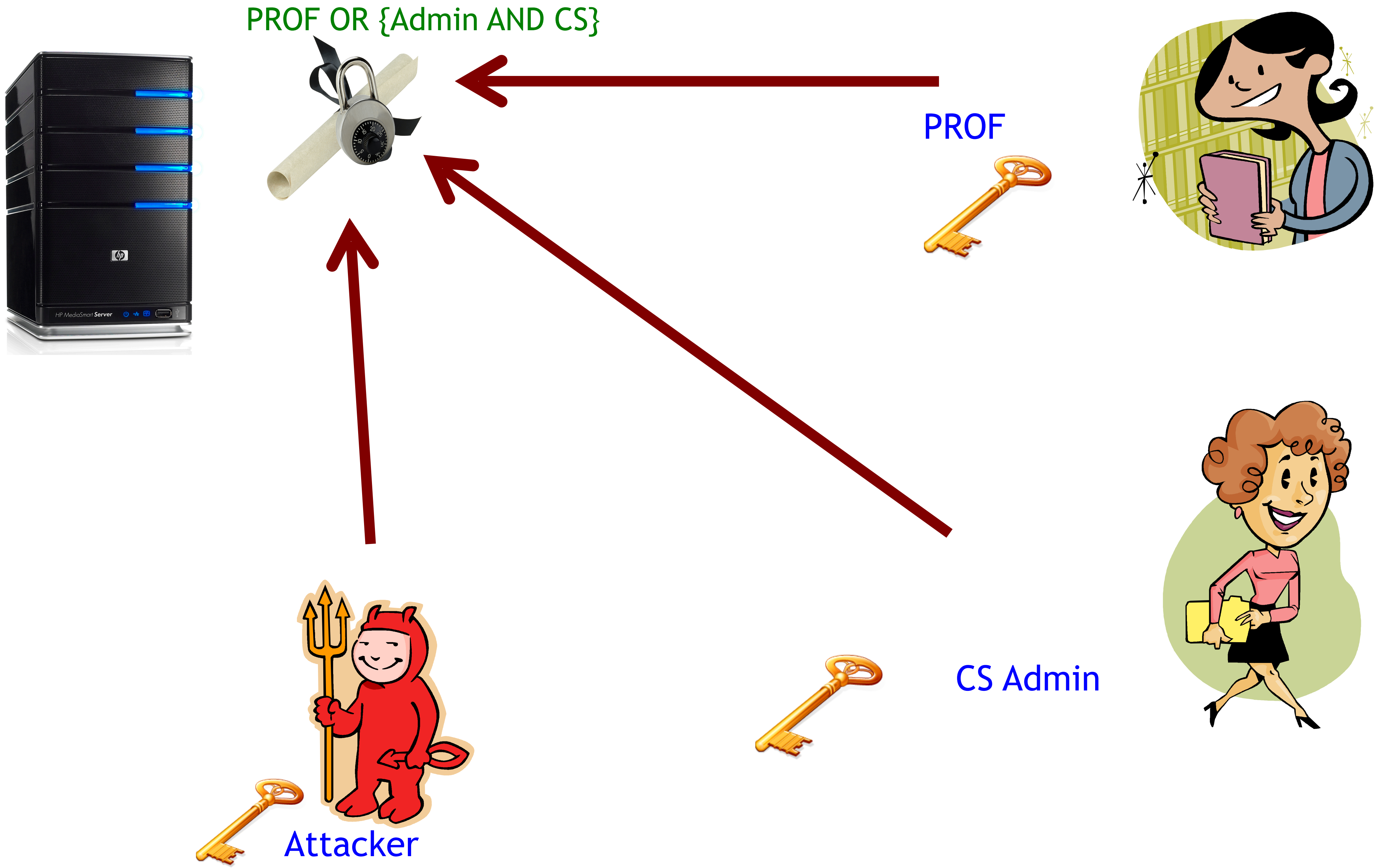
PROF

CS Admin

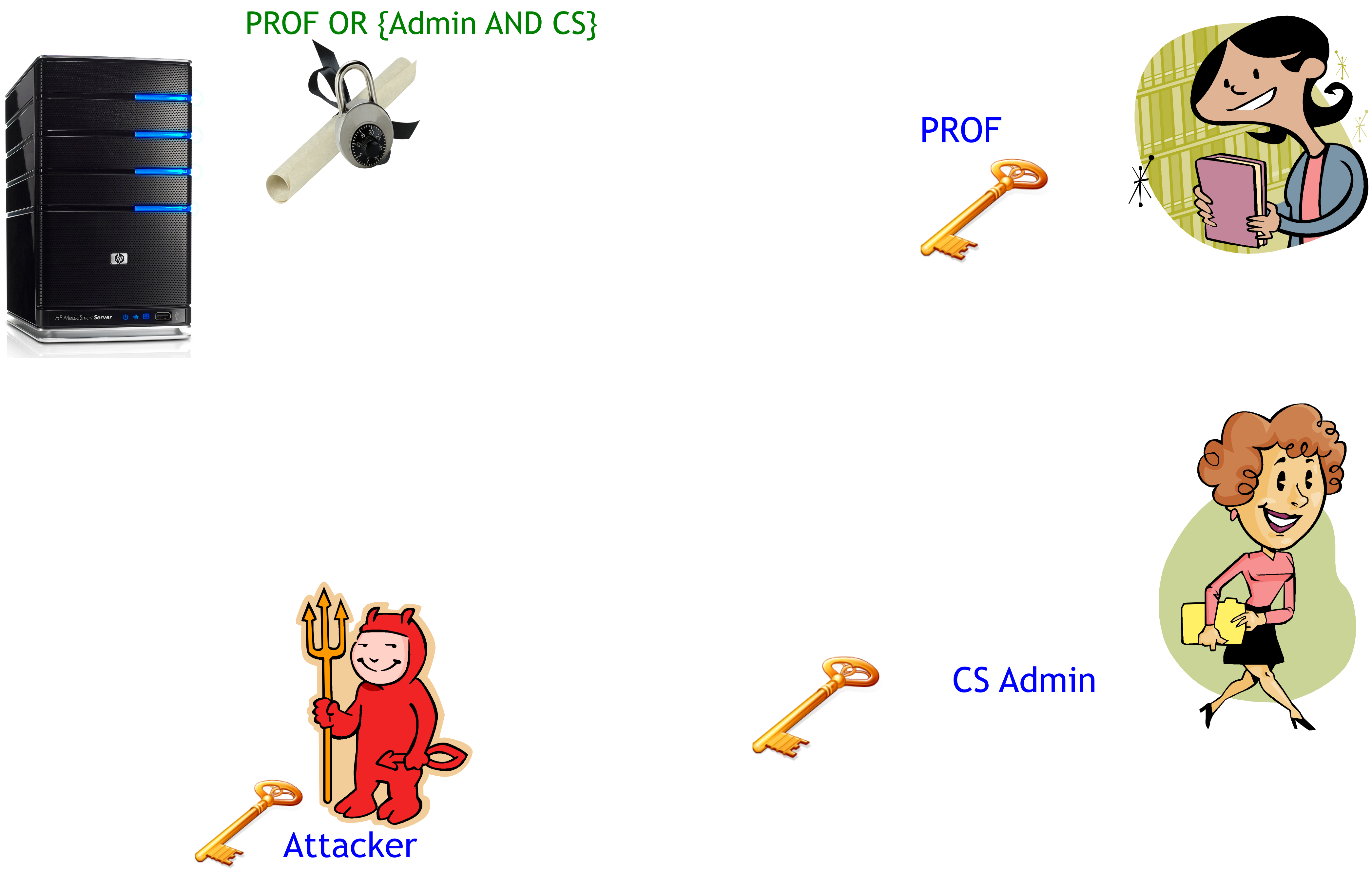
What do we want?



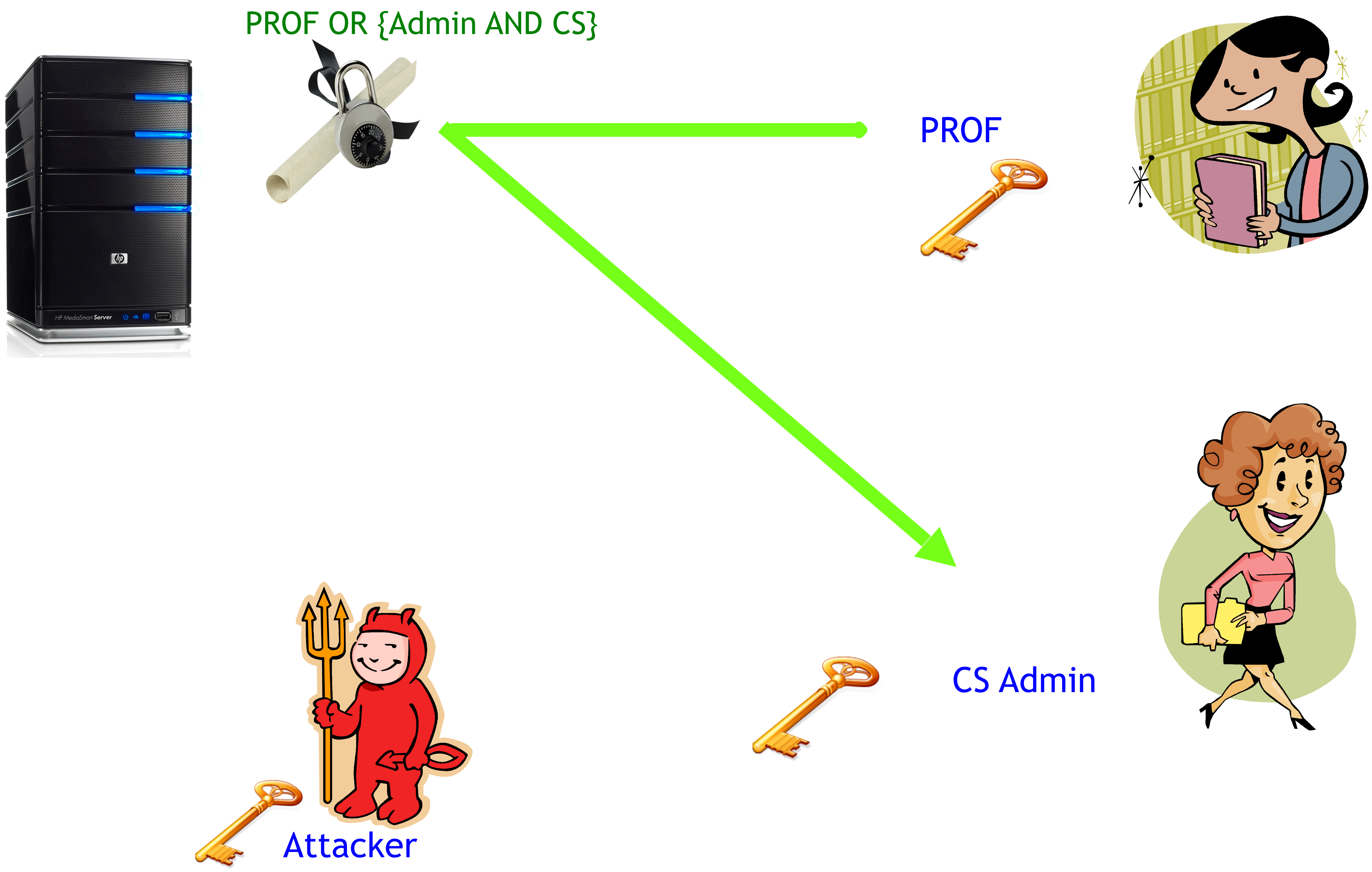
What do we want?



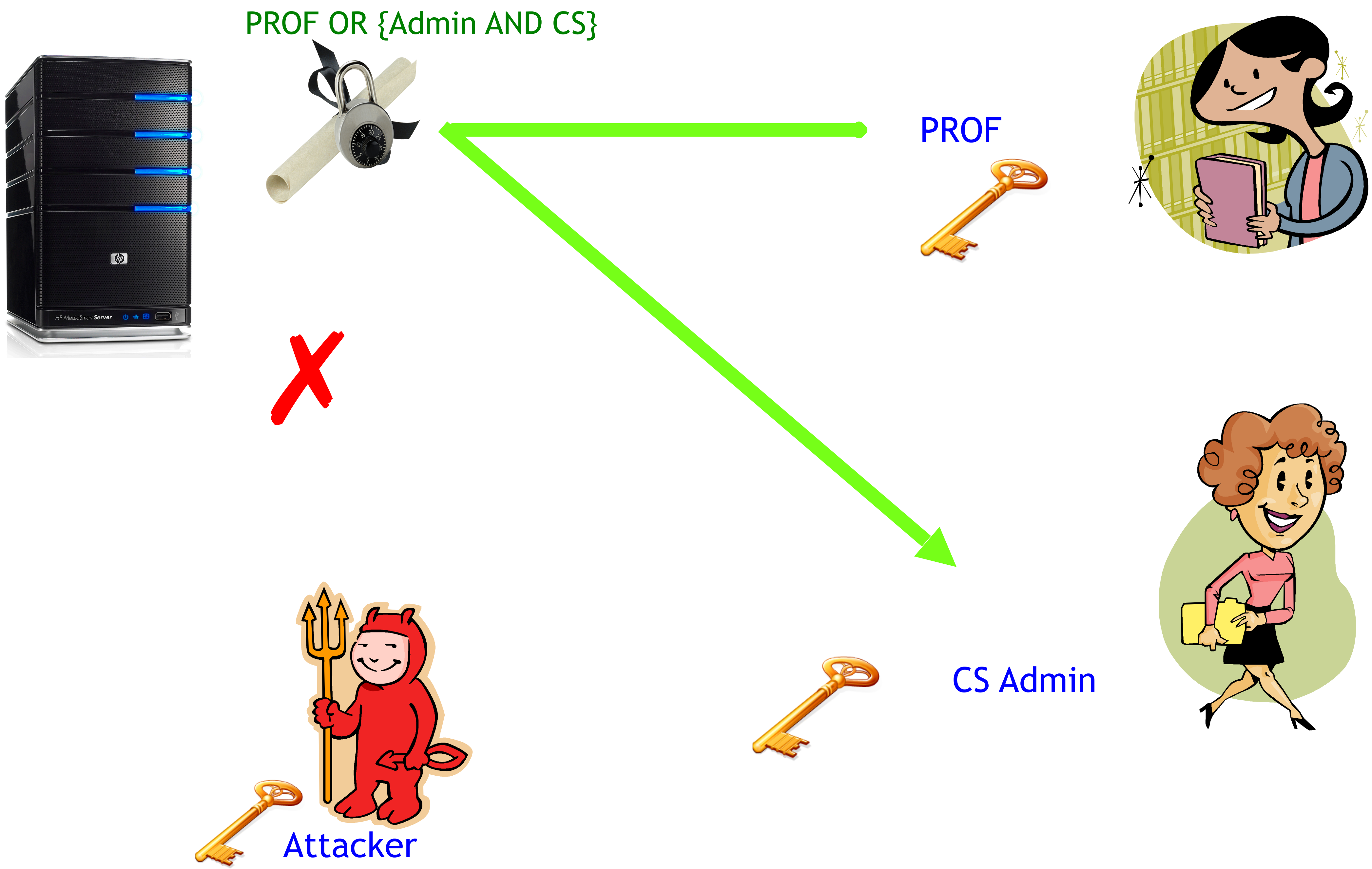
What do we want?



What do we want?



What do we want?




Need New Tools & Techniques!
Main Tool: Lattice Trapdoors

Trapdoor Functions

Trapdoor Functions

Generate (f, T)

Trapdoor Functions

Generate (f, T)

$f : D \to R,$

Trapdoor Functions

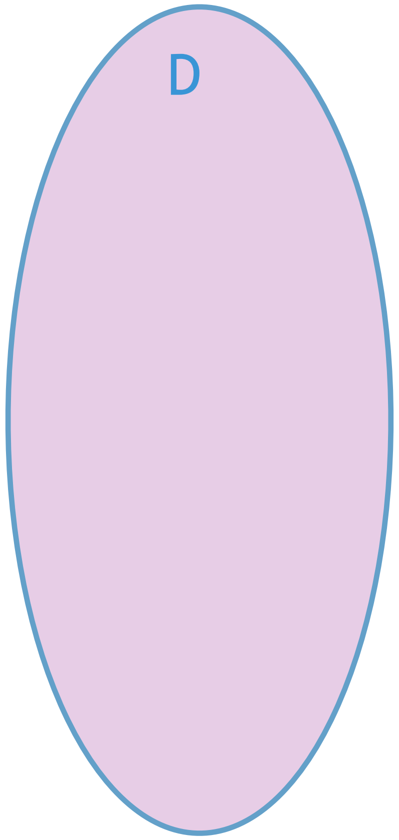
Generate (f, T)

$f : D \rightarrow R$, One Way

Trapdoor Functions

Generate (f, T)

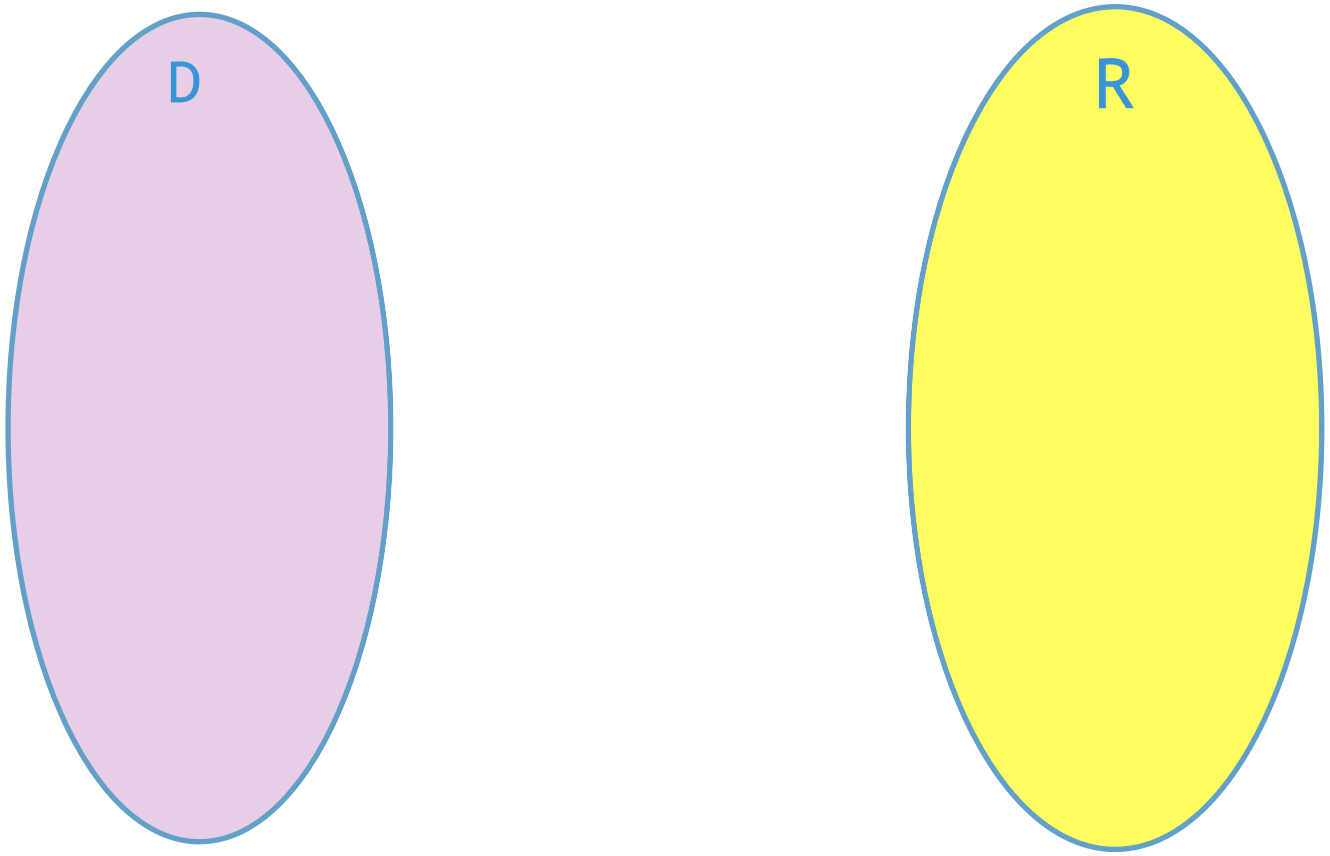
$f : D \to R$, One Way



Trapdoor Functions

Generate (f, T)

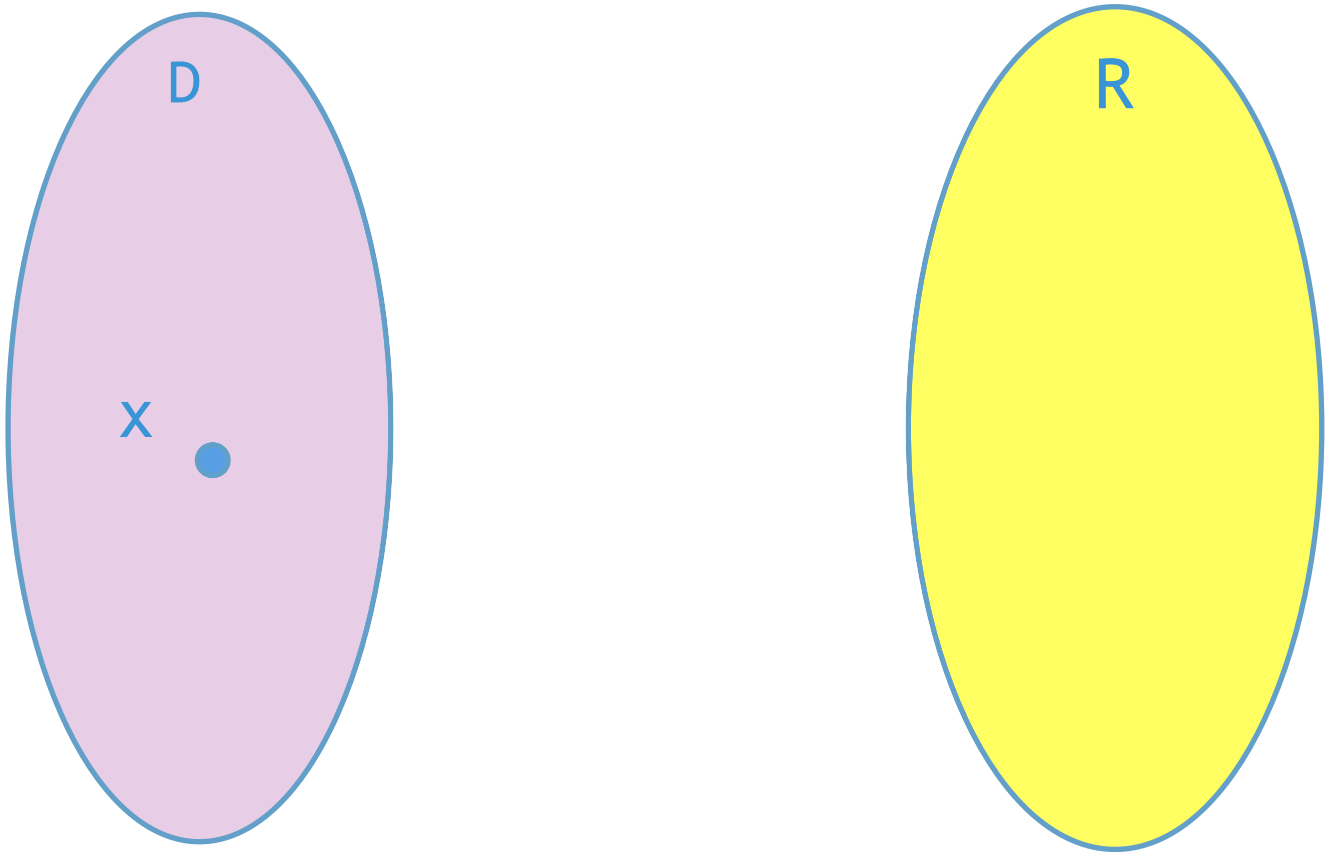
$f : D \to R$, One Way



Trapdoor Functions

Generate (f, T)

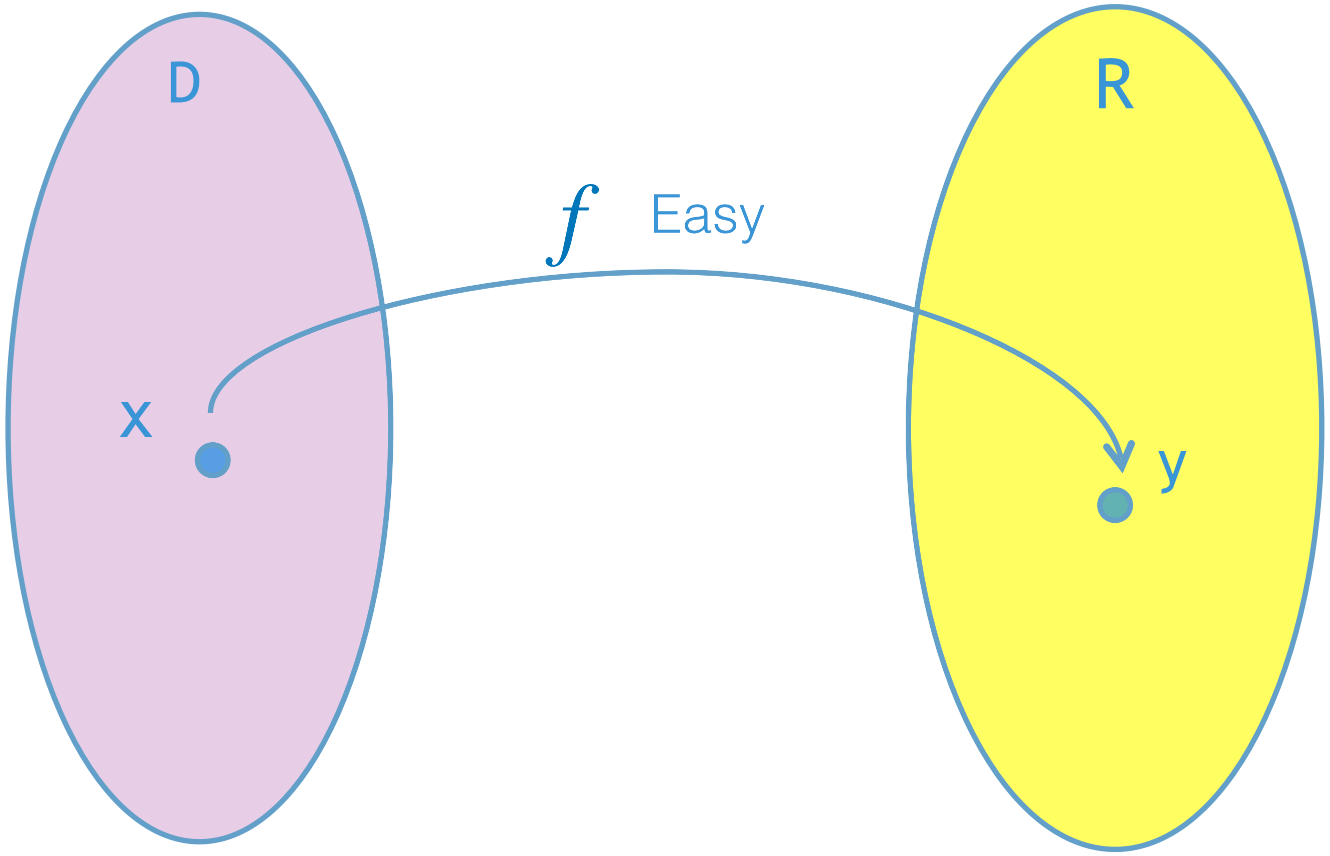
$f : D \to R$, One Way



Trapdoor Functions

Generate (f, T)

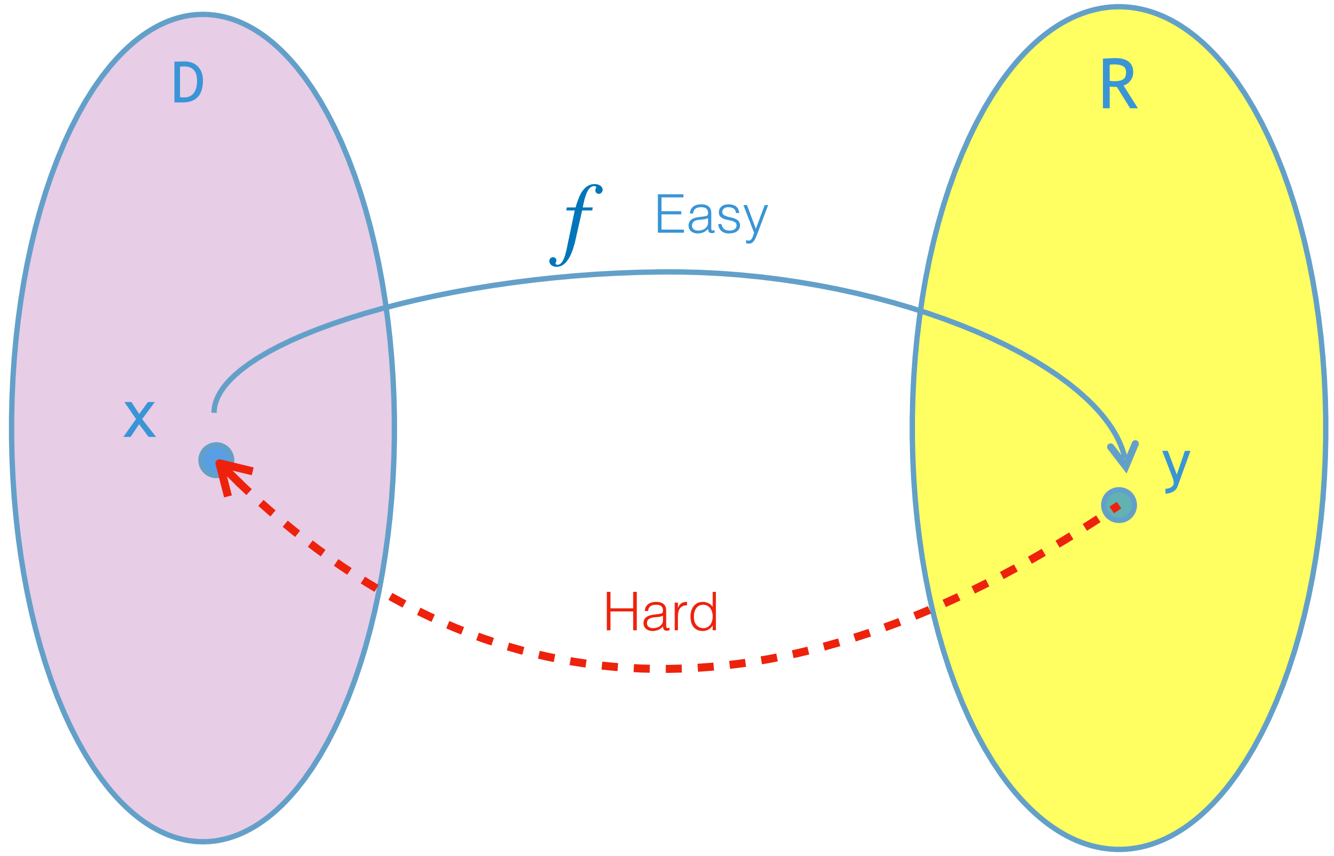
$f : D \to R$, One Way



Trapdoor Functions

Generate (f, T)

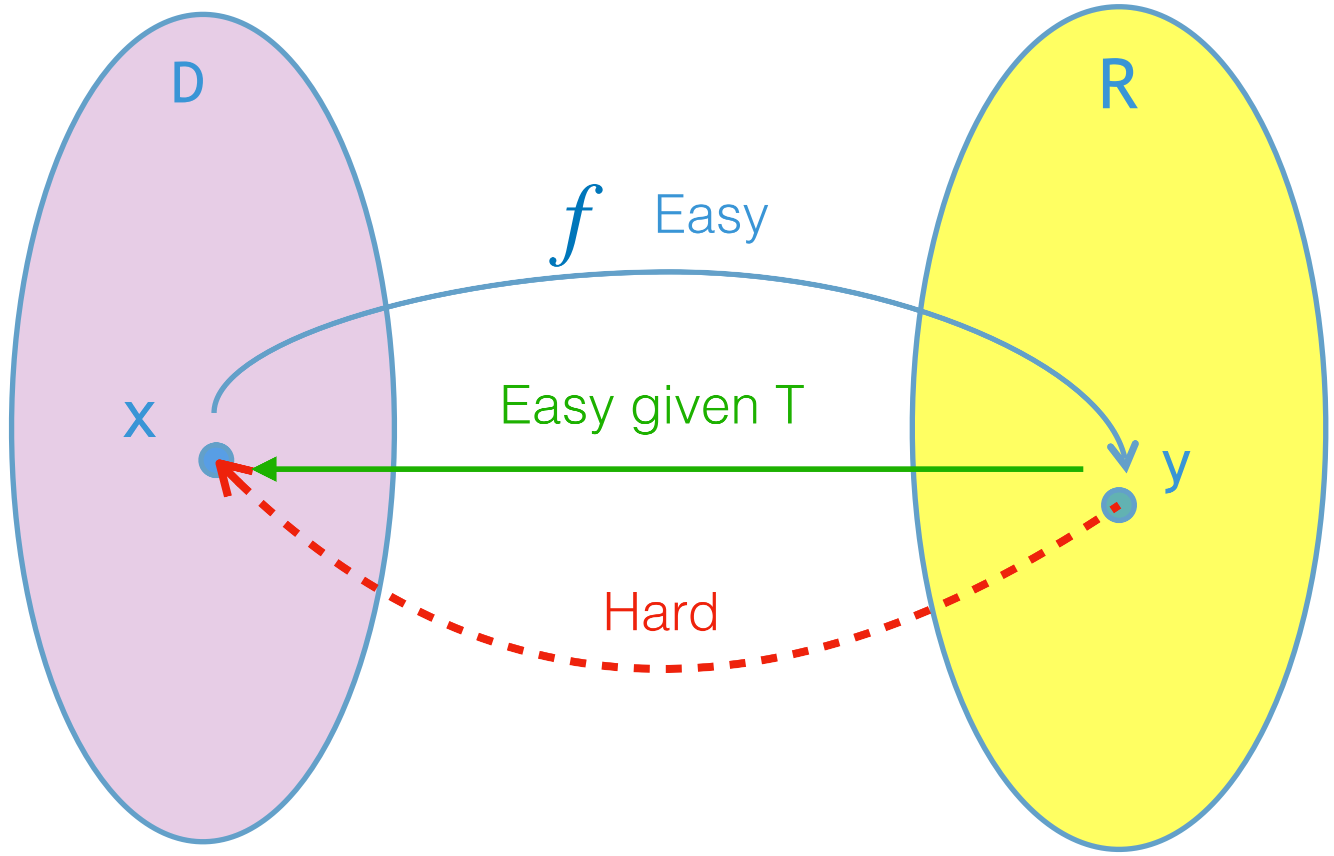
$f : D \to R$, One Way



Trapdoor Functions

Generate (f, T)

$f : D \to R$, One Way

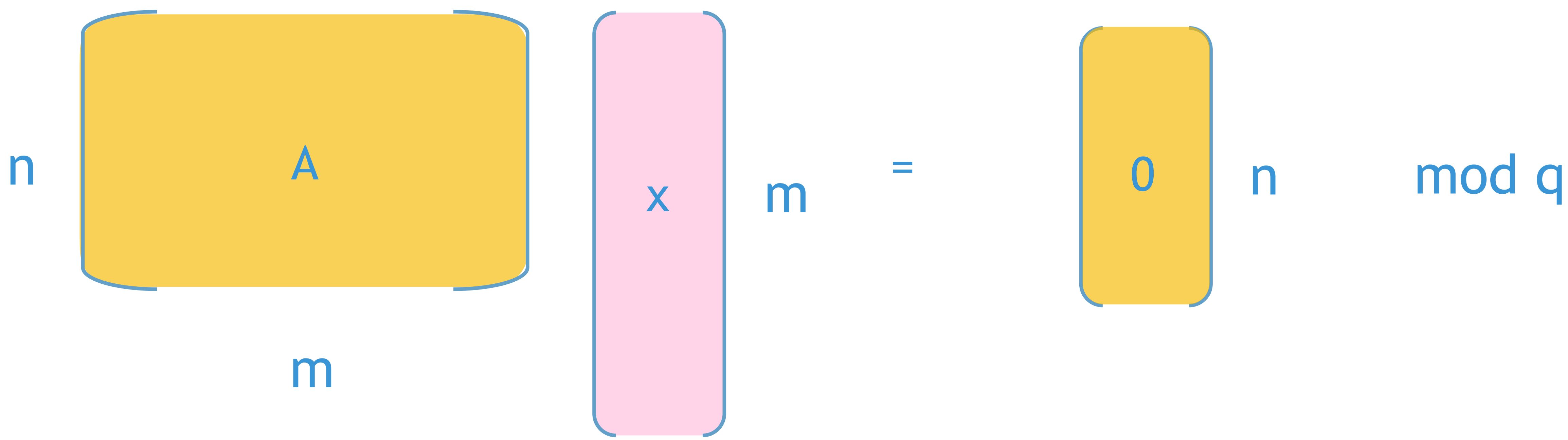


Short Integer Solution Problem

Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $q = \text{poly}(n)$, $m = \Omega(n \log q)$

Given matrix **A**, find "short" (low norm) vector **x** such that

$$\mathbf{A}\,\mathbf{x} = 0 \mod q \;\in \mathbb{Z}_q^n$$



Learning With Errors Problem

Distinguish "noisy inner products" from uniform

Fix uniform $s \in Z_q^n$

a_1 , $b_1 = \langle a_1, s \rangle + e_1$
a_2 , $b_2 = \langle a_2, s \rangle + e_2$
$\vdots$
a_m , $b_m = \langle a_m, s \rangle + e_m$

vs

a'_1 , b'_1
a'_2 , b'_2
$\vdots$
a'_m , b'_m

a_i uniform $\in Z_q^n$, $e_i \sim \phi \in Z_q$

a_i uniform $\in Z_q^n$, b_i uniform $\in Z_q$

Lattice Based One Way Functions

Public Key $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $q = \text{poly}(n)$, $m = \Omega(n \log q)$

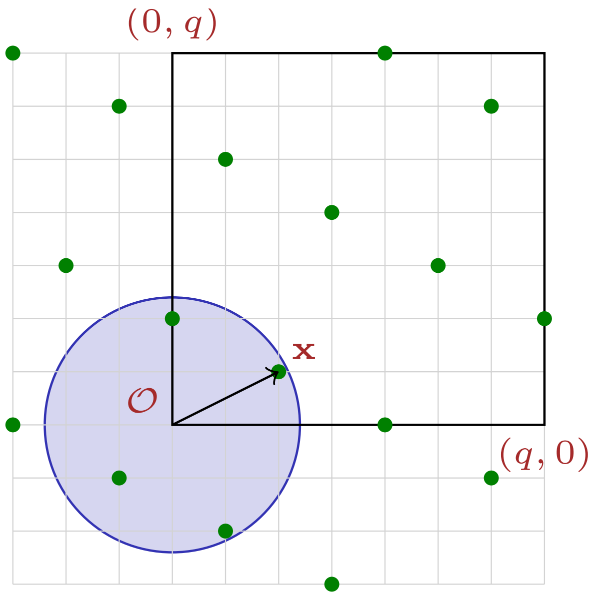
Lattice Based One Way Functions

Public Key $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $q = \text{poly}(n)$, $m = \Omega(n \log q)$

Based on SIS

$$f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\,\mathbf{x} \mod q \ \in \mathbb{Z}_q^n$$

- Short x, surjective
- CRHF if SIS is hard [Ajt96…]



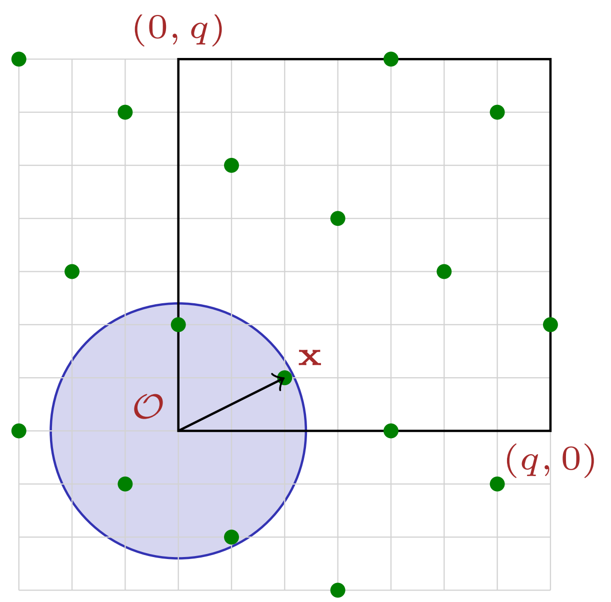
Lattice Based One Way Functions

Public Key $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, $q = \text{poly}(n)$, $m = \Omega(n \log q)$

Based on SIS

$$f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\,\mathbf{x} \mod q \in \mathbb{Z}_q^n$$

- Short x, surjective
- CRHF if SIS is hard [Ajt96…]



Based on LWE

$$g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \mod q \in \mathbb{Z}_q^m$$

- Very short e, injective
- OWF if LWE is hard [Reg05…]

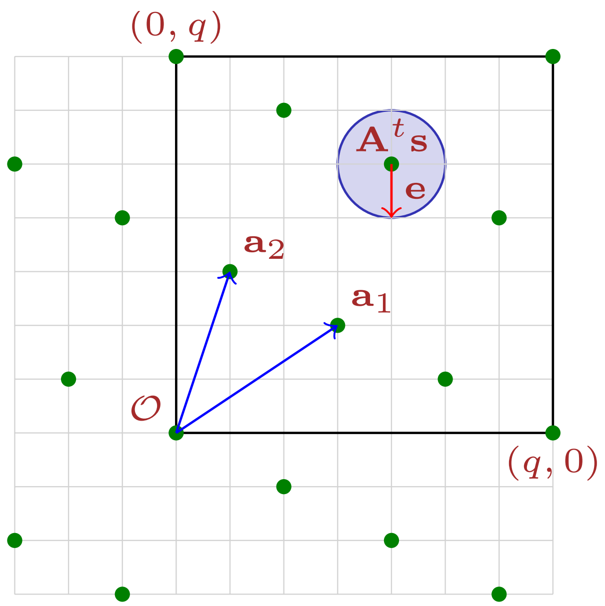


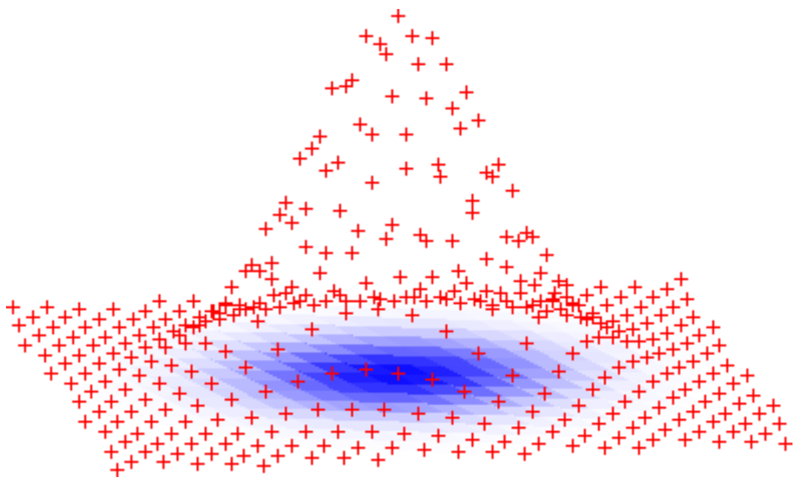
Image Credit: MP12 slides

Inverting functions for Crypto

- Given $\mathbf{u} = f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\ \mathbf{x} \mod q$
- Sample

$$\mathbf{x}' \leftarrow = f_{\mathbf{A}}^{-1}(\mathbf{u})$$

with prob $\propto \exp(-\|\mathbf{x}'\|^2/\sigma^2)$



And

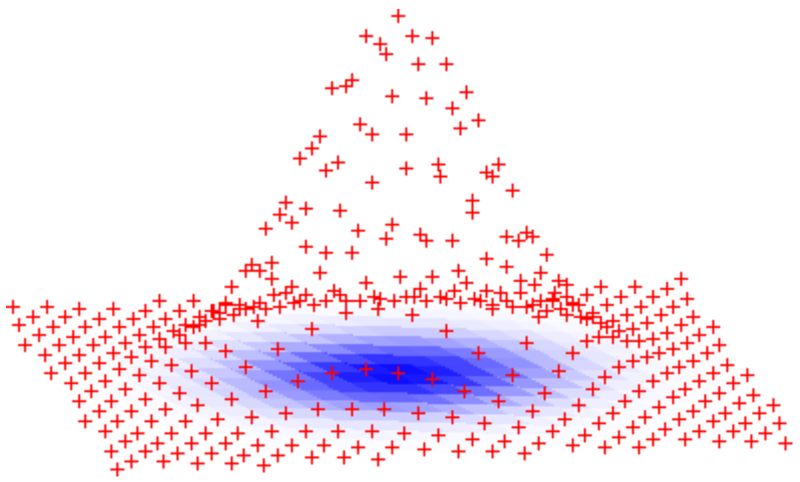
- Given $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \mod q$
- Find unique $(\mathbf{s}, \mathbf{e})$

Inverting functions for Crypto

- Given $\mathbf{u} = f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\ \mathbf{x} \mod q$
- Sample

$$\mathbf{x}' \leftarrow = f_{\mathbf{A}}^{-1}(\mathbf{u})$$

with prob $\propto \exp(-\|\mathbf{x}'\|^2/\sigma^2)$

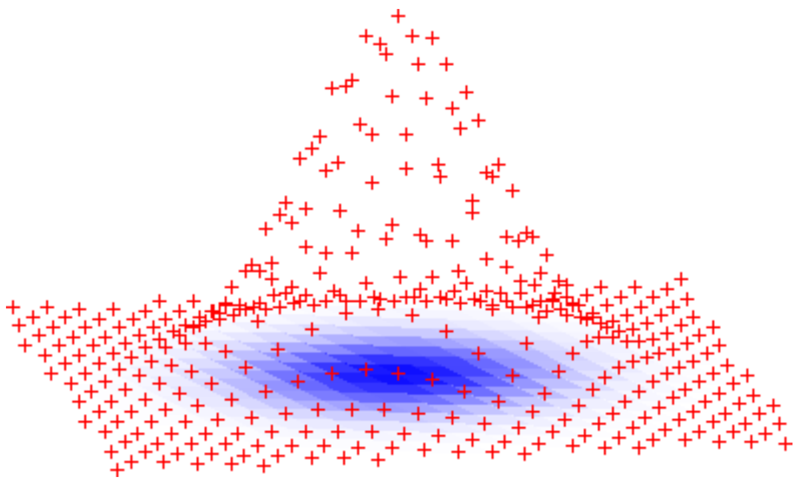


Inverting functions for Crypto

- Given $\mathbf{u} = f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\,\mathbf{x} \mod q$
- Sample

$$\mathbf{x}' \leftarrow = f_{\mathbf{A}}^{-1}(\mathbf{u})$$

with prob $\propto \exp(-\|\mathbf{x}'\|^2/\sigma^2)$



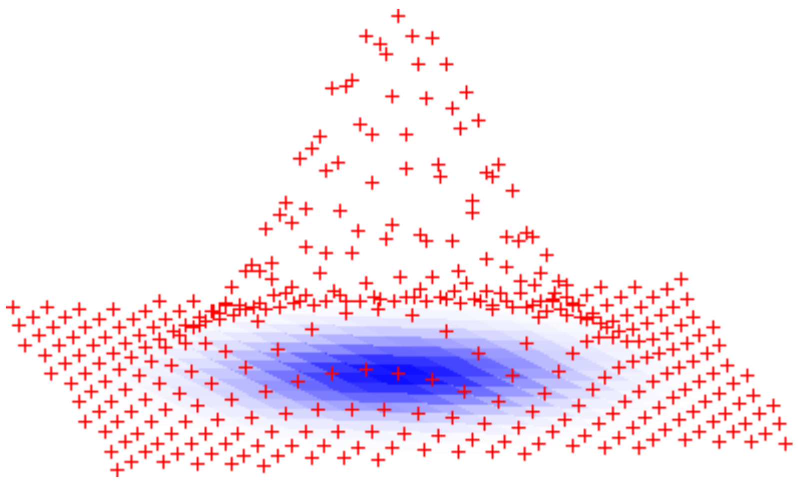
Preimage Sampleable Trapdoor Functions!

Inverting functions for Crypto

- Given $\mathbf{u} = f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\ \mathbf{x} \mod q$
- Sample

$$\mathbf{x}' \leftarrow = f_{\mathbf{A}}^{-1}(\mathbf{u})$$

with prob $\propto \exp(-\|\mathbf{x}'\|^2/\sigma^2)$

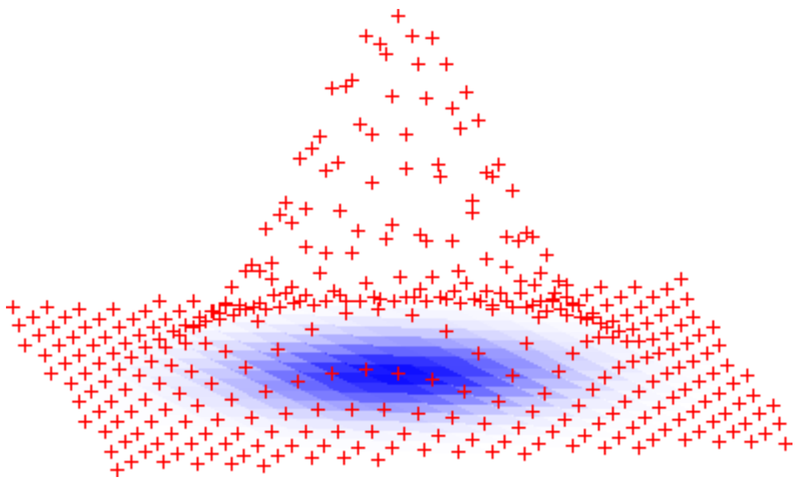


Preimage Sampleable Trapdoor Functions!

Generate (x, y) in two equivalent ways

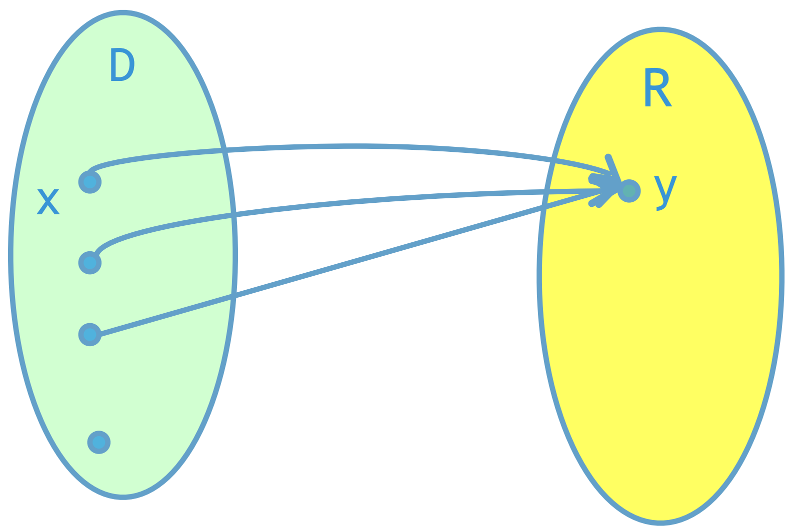
Inverting functions for Crypto

- Given $\mathbf{u} = f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\ \mathbf{x} \mod q$
- Sample
  $$\mathbf{x}' \leftarrow = f_{\mathbf{A}}^{-1}(\mathbf{u})$$
  with prob $\propto \exp(-\|\mathbf{x}'\|^2/\sigma^2)$



Preimage Sampleable Trapdoor Functions!

Generate (x, y) in two equivalent ways

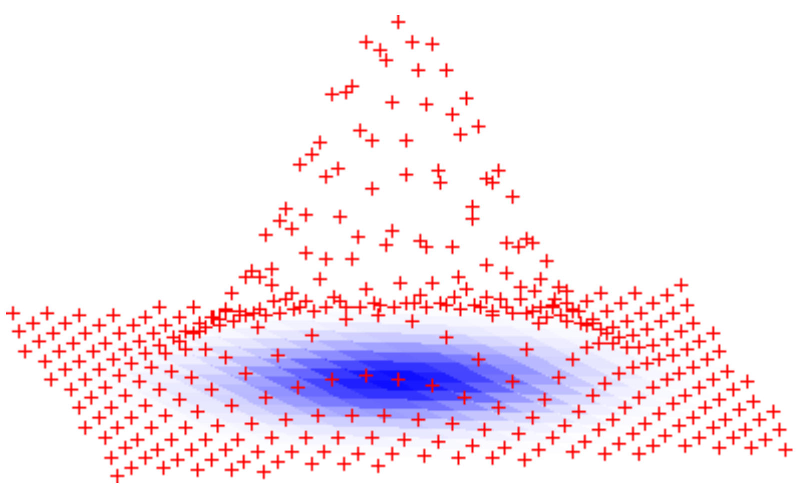


Inverting functions for Crypto

- Given $\mathbf{u} = f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\ \mathbf{x} \mod q$
- Sample

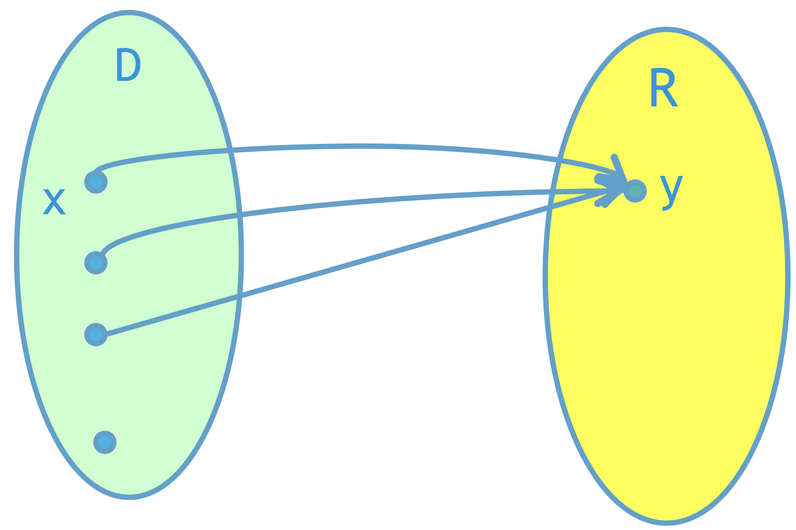
$$\mathbf{x}' \leftarrow = f_{\mathbf{A}}^{-1}(\mathbf{u})$$

with prob $\propto \exp(-\|\mathbf{x}'\|^2/\sigma^2)$



Preimage Sampleable Trapdoor Functions!

Generate (x, y) in two equivalent ways



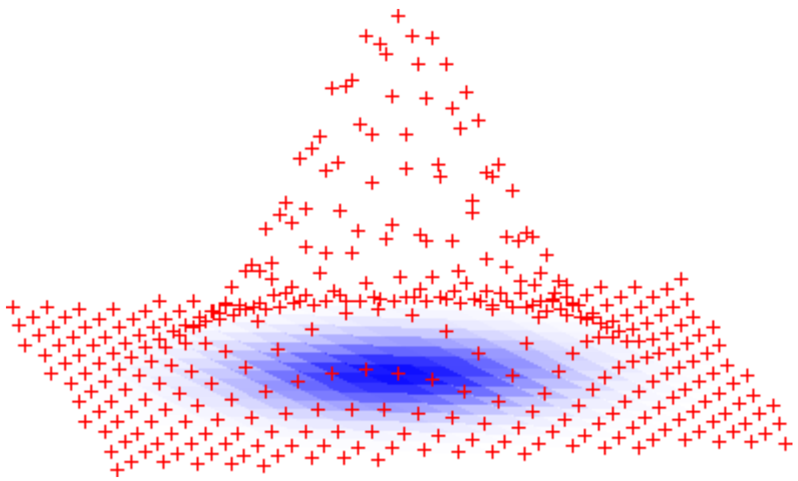
OR

Inverting functions for Crypto

- Given $\mathbf{u} = f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\ \mathbf{x} \mod q$
- Sample

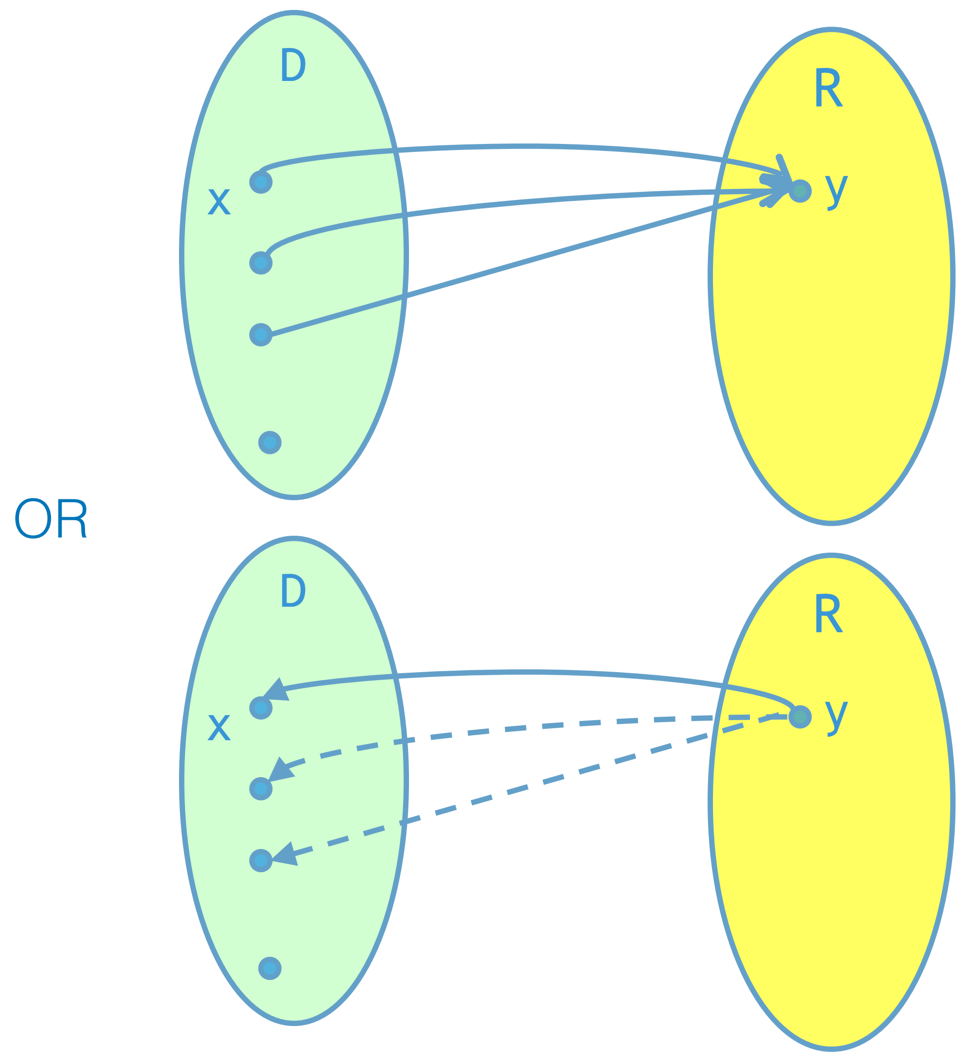
$$\mathbf{x}' \leftarrow = f_{\mathbf{A}}^{-1}(\mathbf{u})$$

with prob $\propto \exp(-\|\mathbf{x}'\|^2/\sigma^2)$



Preimage Sampleable Trapdoor Functions!

Generate (x, y) in two equivalent ways

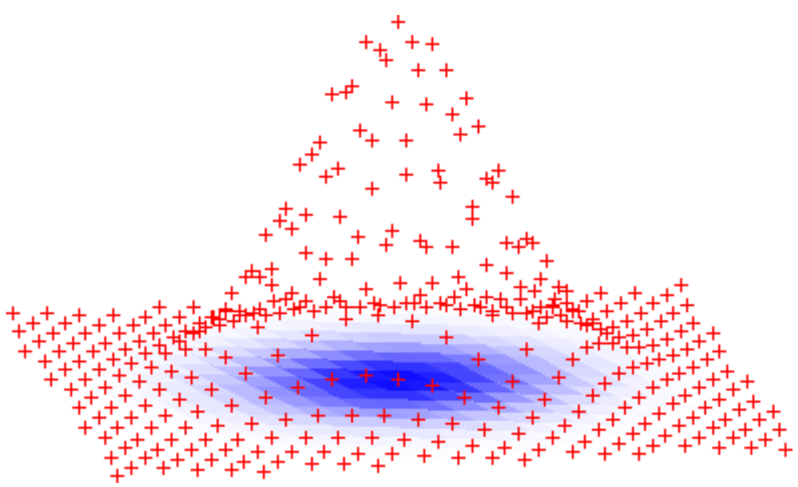


Inverting functions for Crypto

- Given $\mathbf{u} = f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\ \mathbf{x} \mod q$
- Sample

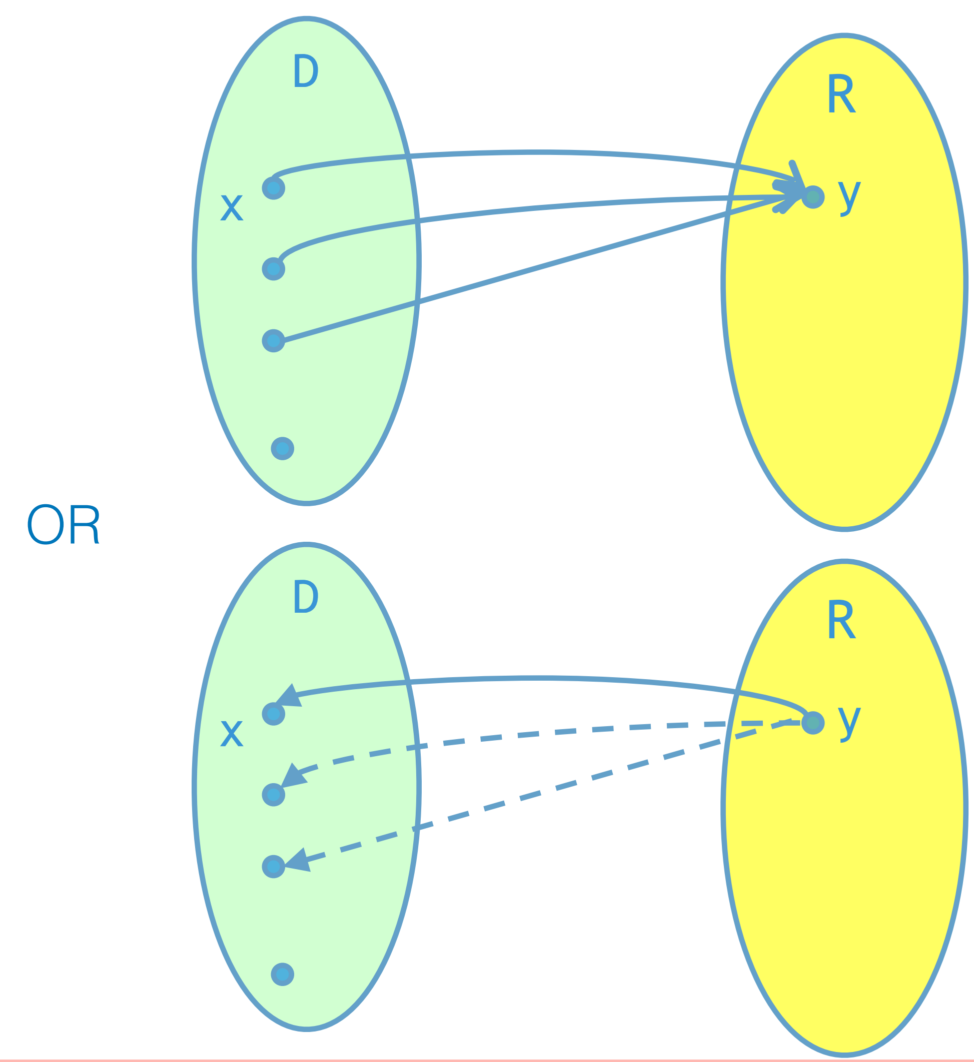
$$\mathbf{x}' \leftarrow = f_{\mathbf{A}}^{-1}(\mathbf{u})$$

with prob $\propto \exp(-\|\mathbf{x}'\|^2/\sigma^2)$



Preimage Sampleable Trapdoor Functions!

Generate (x, y) in two equivalent ways



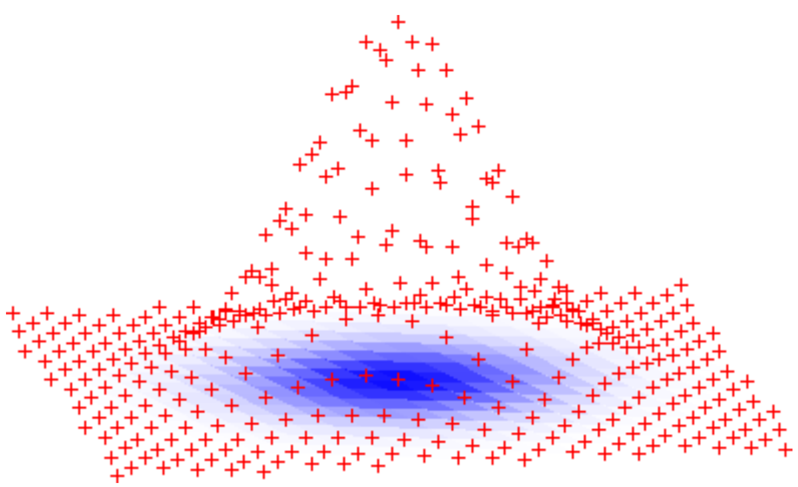
Same Distribution (Discrete Gaussian, Uniform) !

Inverting functions for Crypto

- Given $\mathbf{u} = f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\ \mathbf{x} \mod q$
- Sample

$$\mathbf{x}' \leftarrow = f_{\mathbf{A}}^{-1}(\mathbf{u})$$

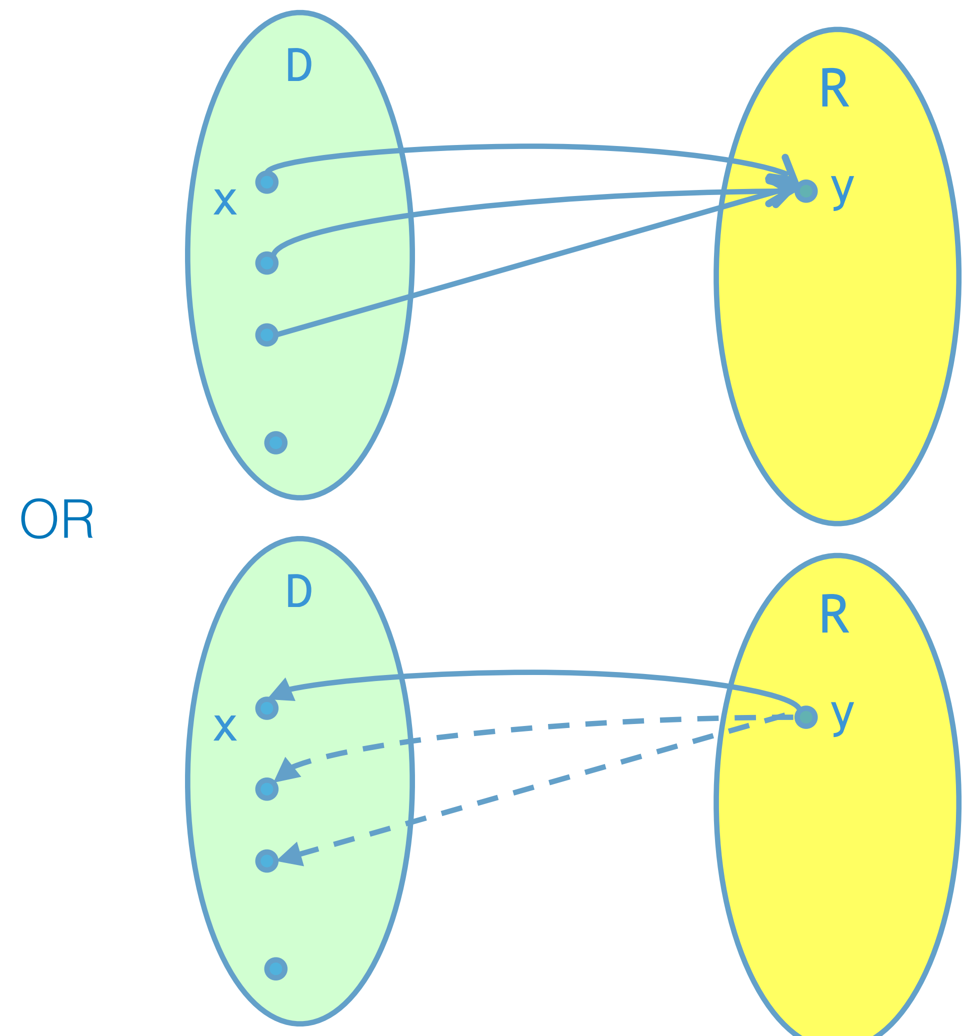
with prob $\propto \exp(-\|\mathbf{x}'\|^2/\sigma^2)$



Latter distribution needs lattice trapdoors!

Preimage Sampleable Trapdoor Functions!

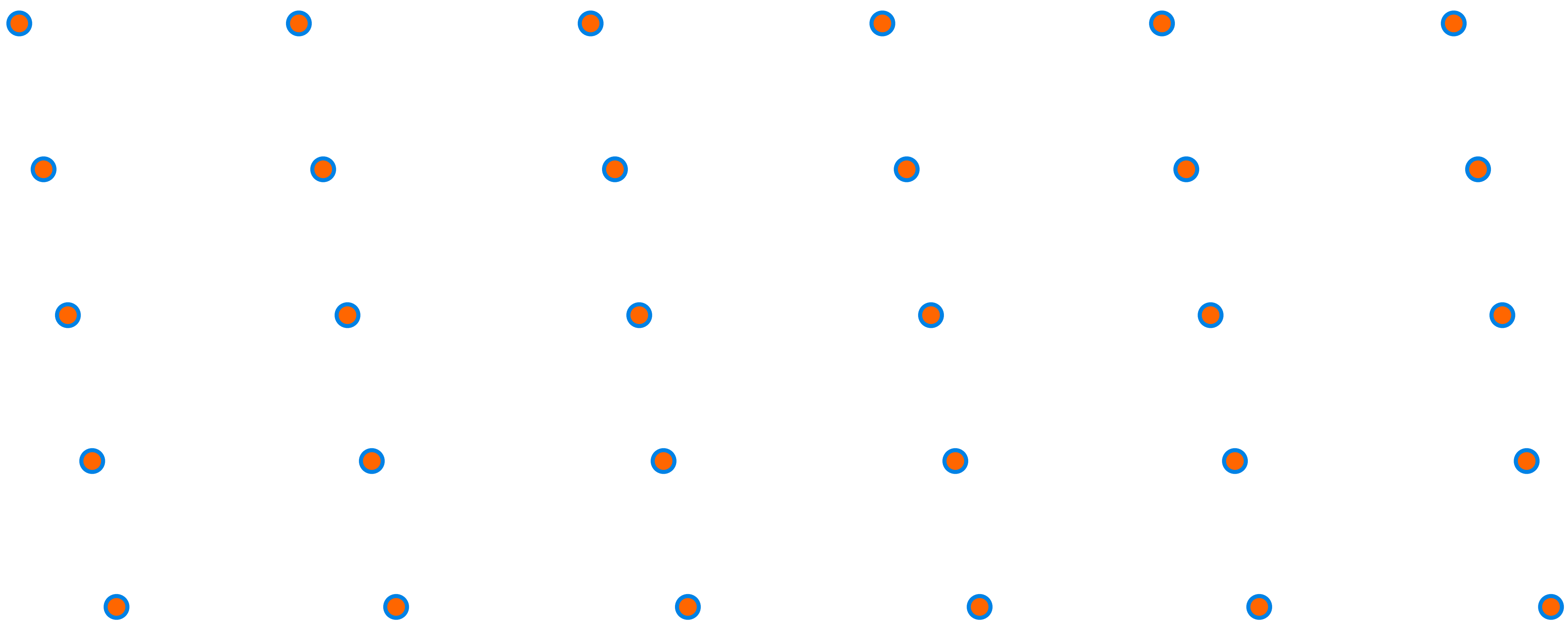
Generate (x, y) in two equivalent ways



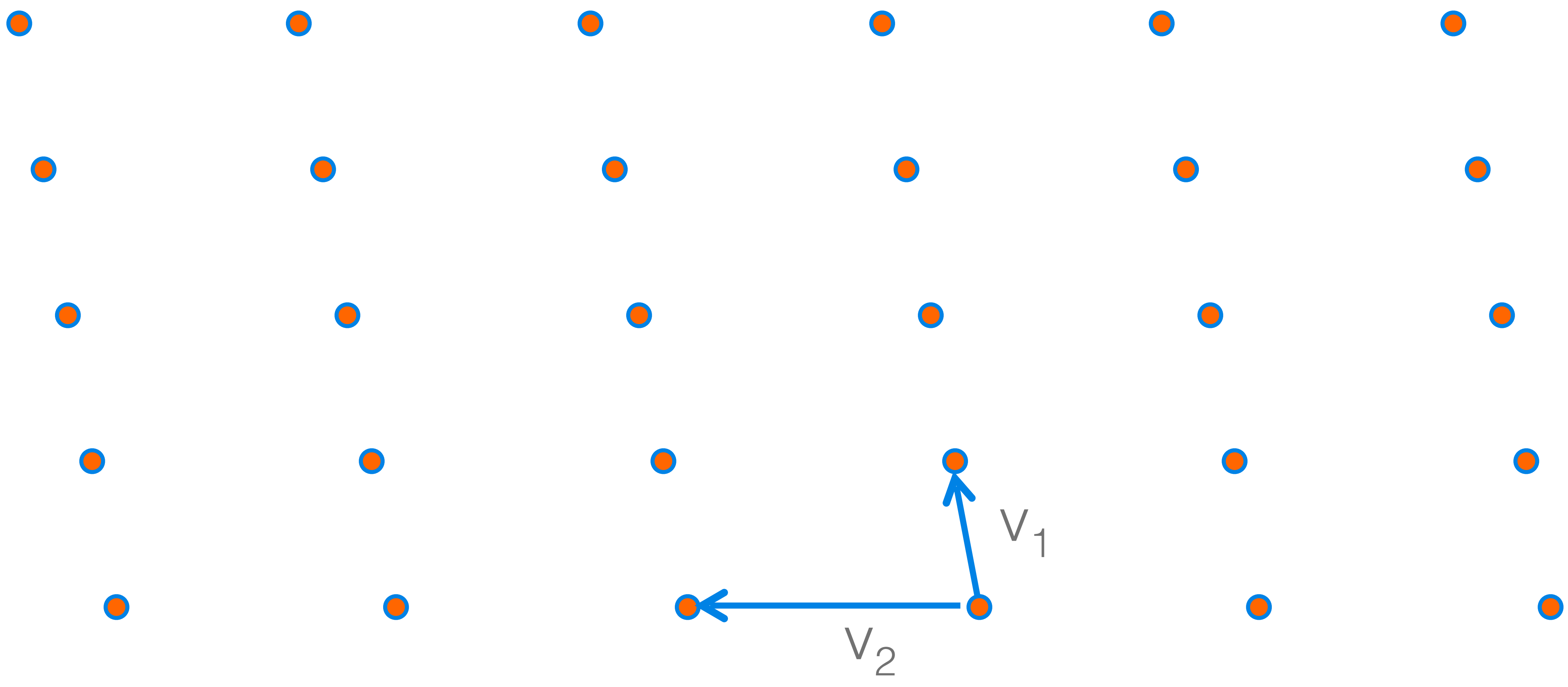
Same Distribution (Discrete Gaussian, Uniform) !

What do these trapdoors look like?

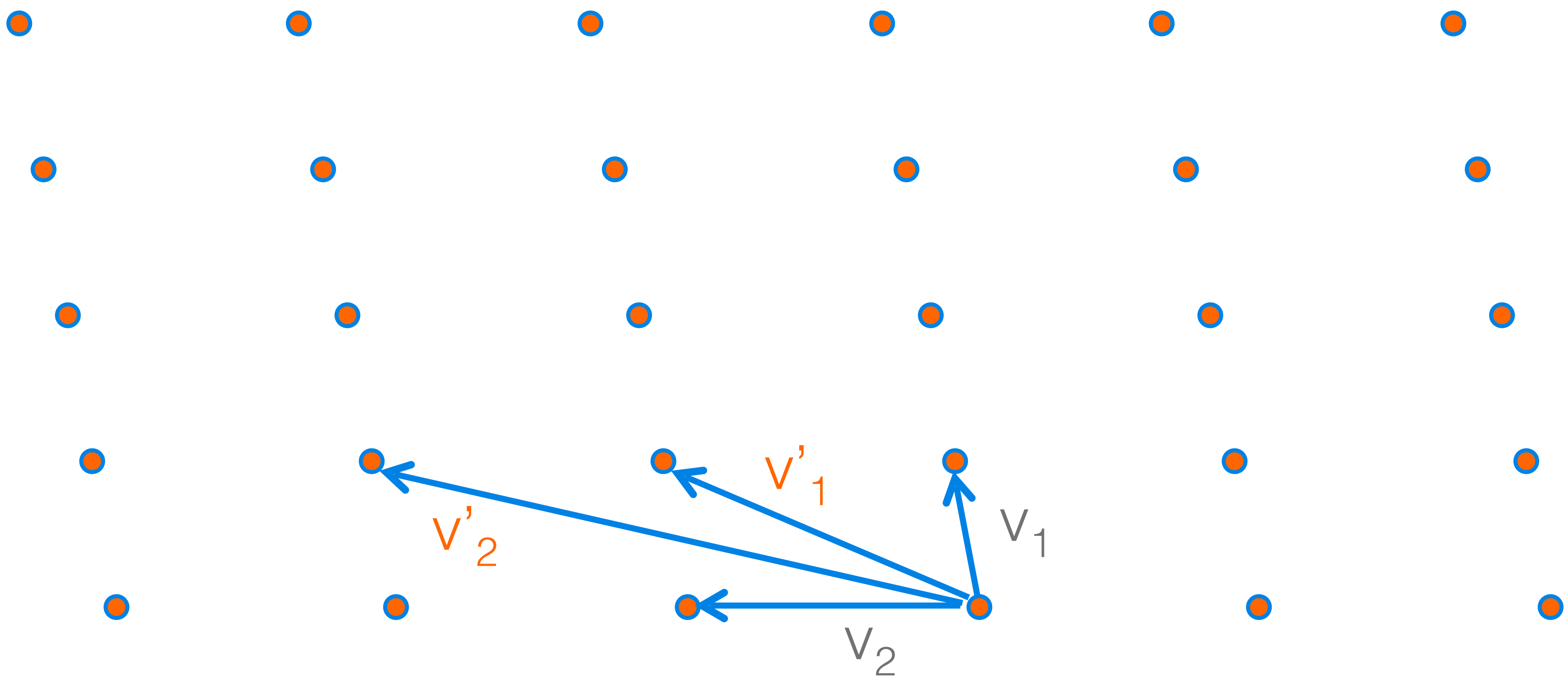
Lattice Trapdoors (Type 1): Geometric View



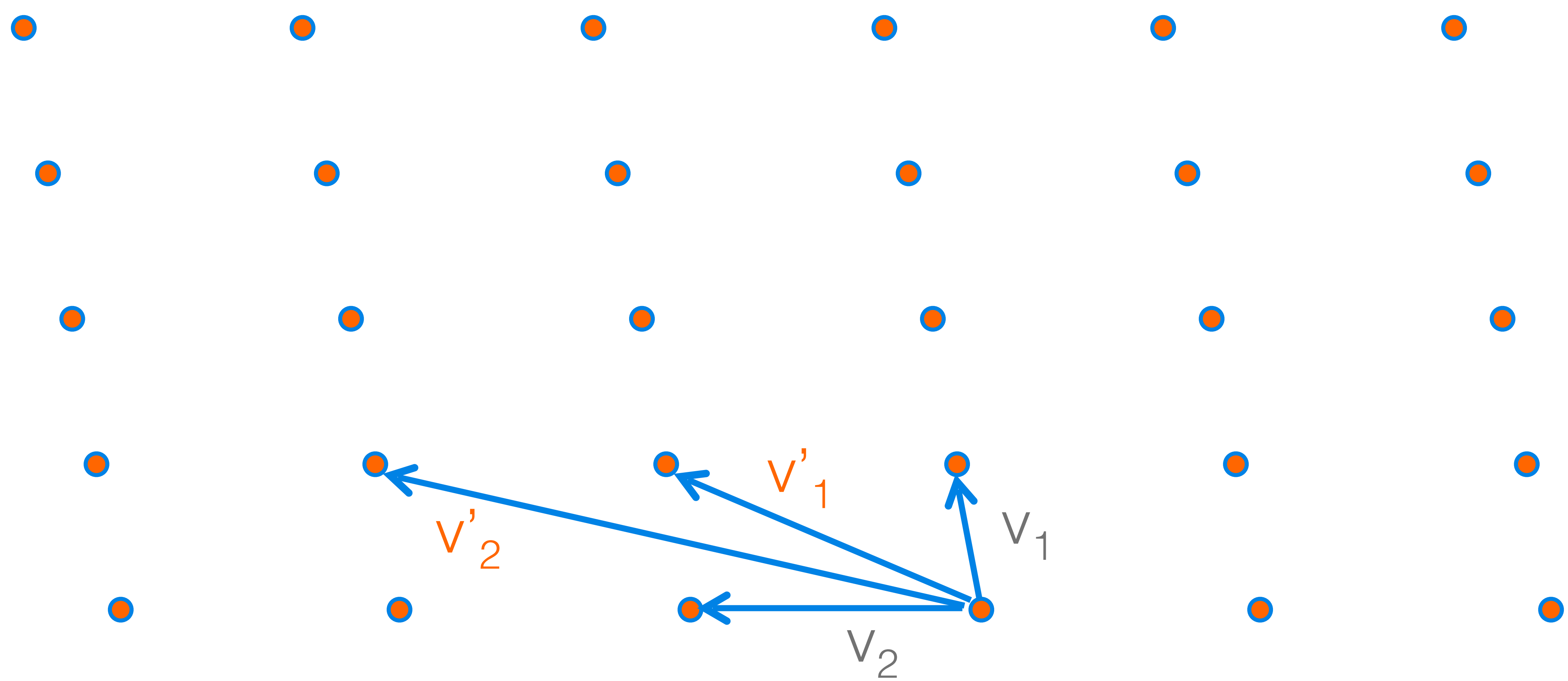
Lattice Trapdoors (Type 1): Geometric View



Lattice Trapdoors (Type 1): Geometric View

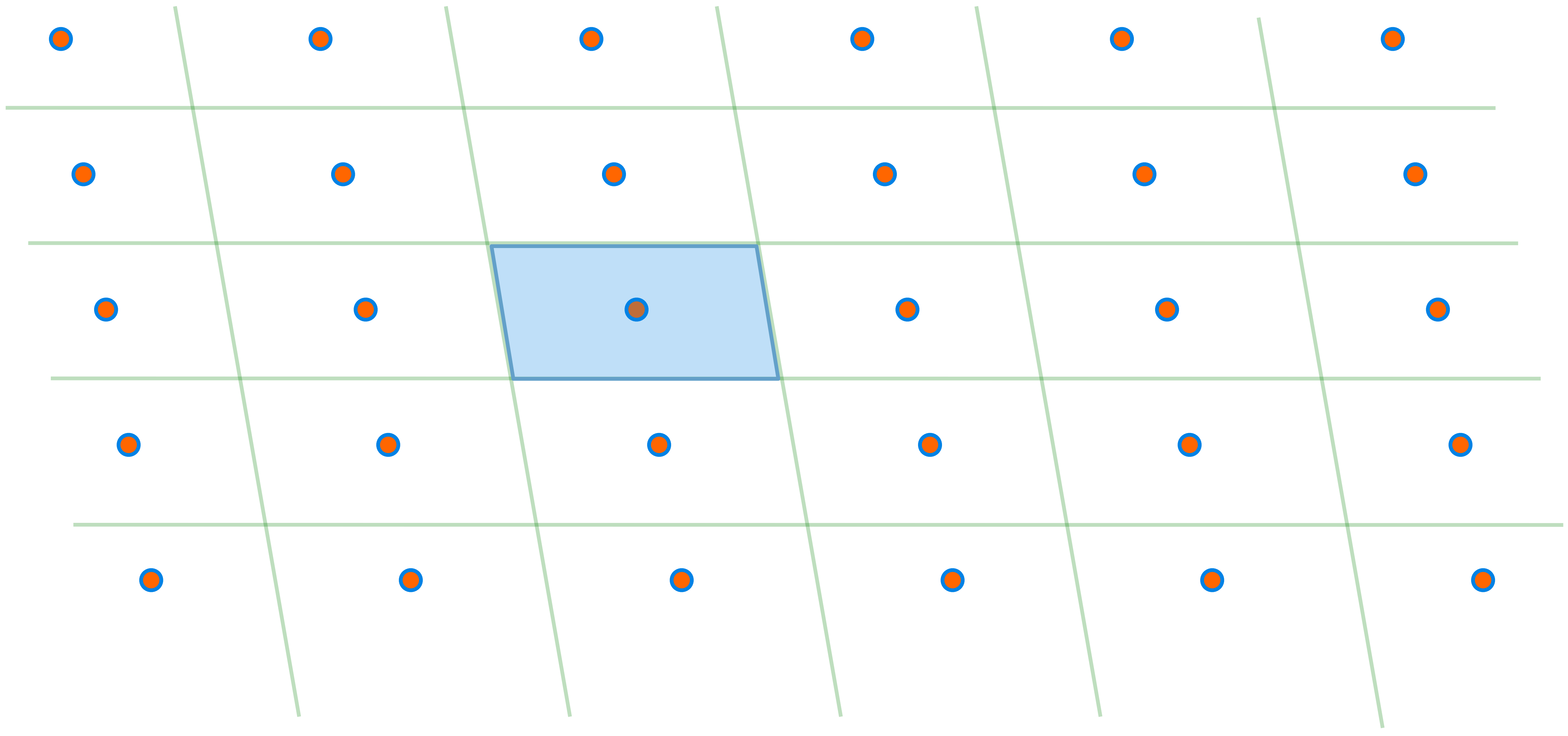


Lattice Trapdoors (Type 1): Geometric View

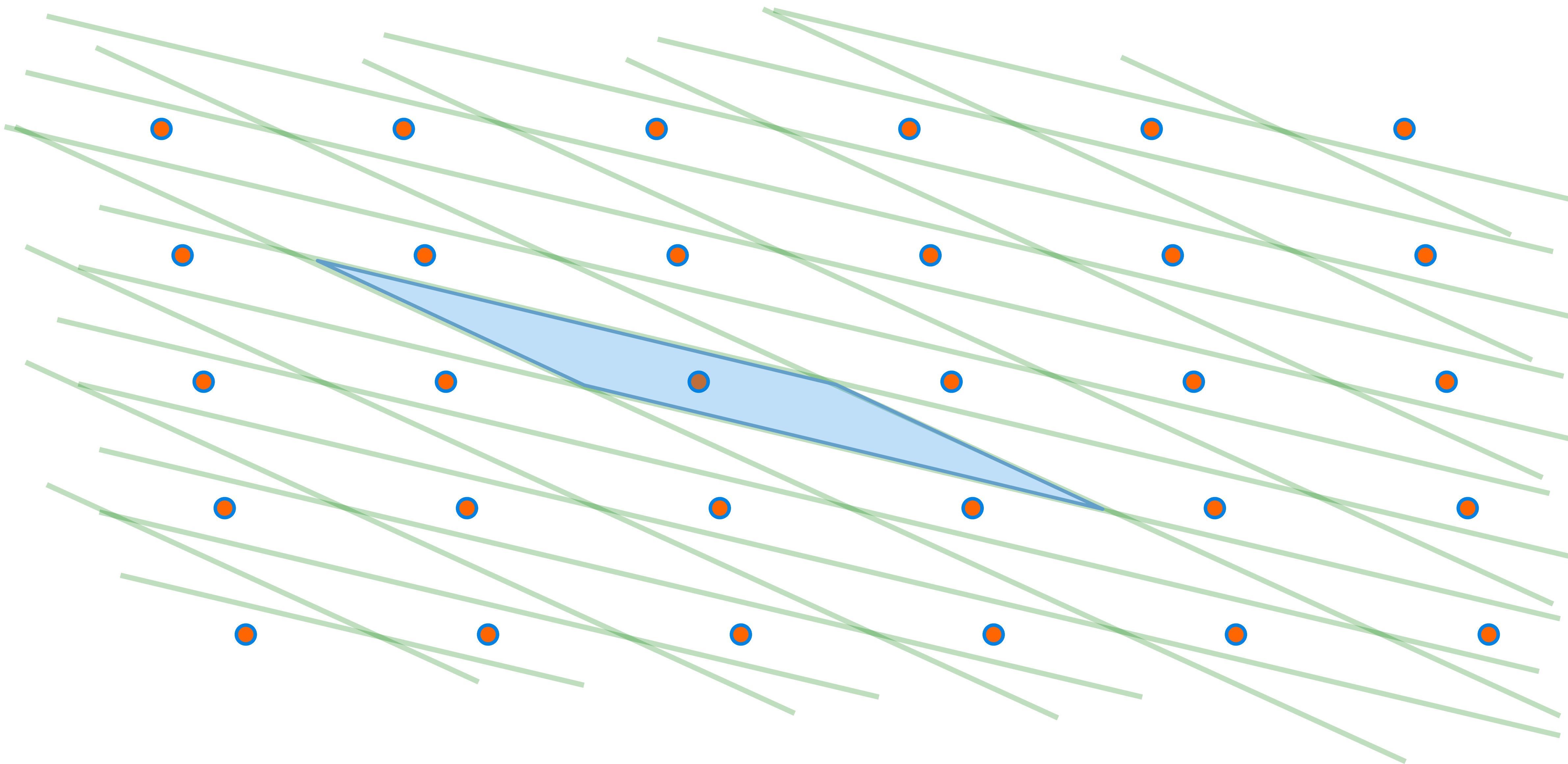


Multiple Bases

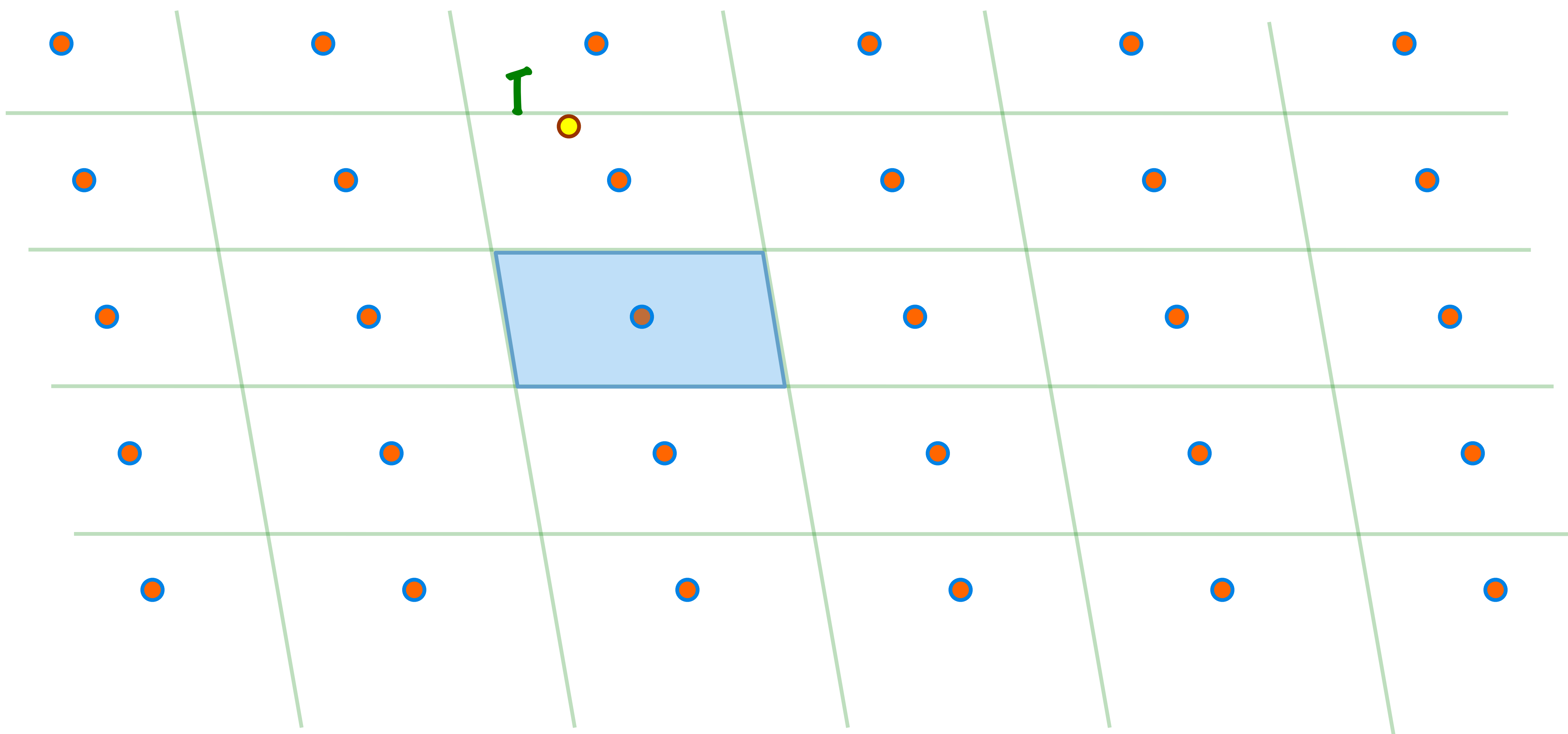
Parallelopipeds



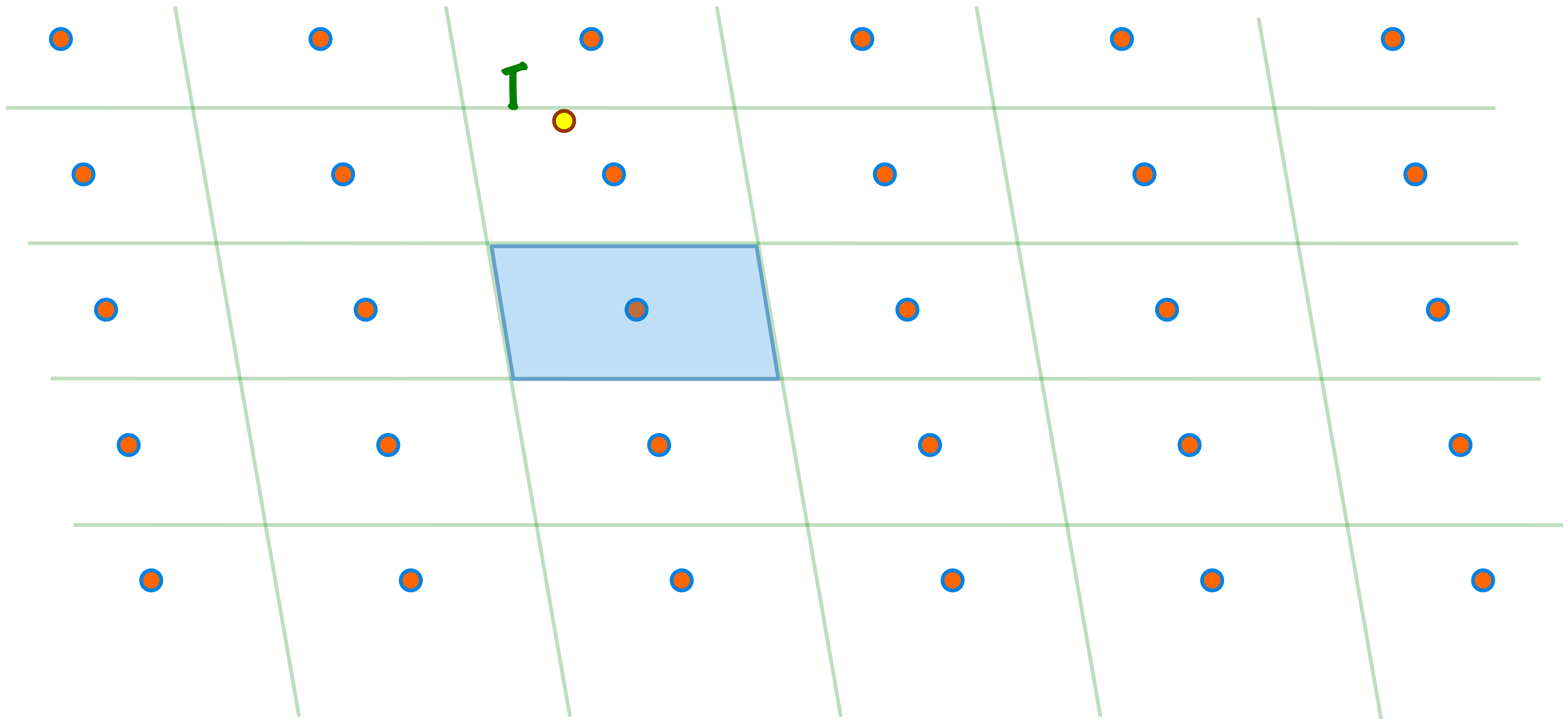
Parallelopipeds



Good Basis

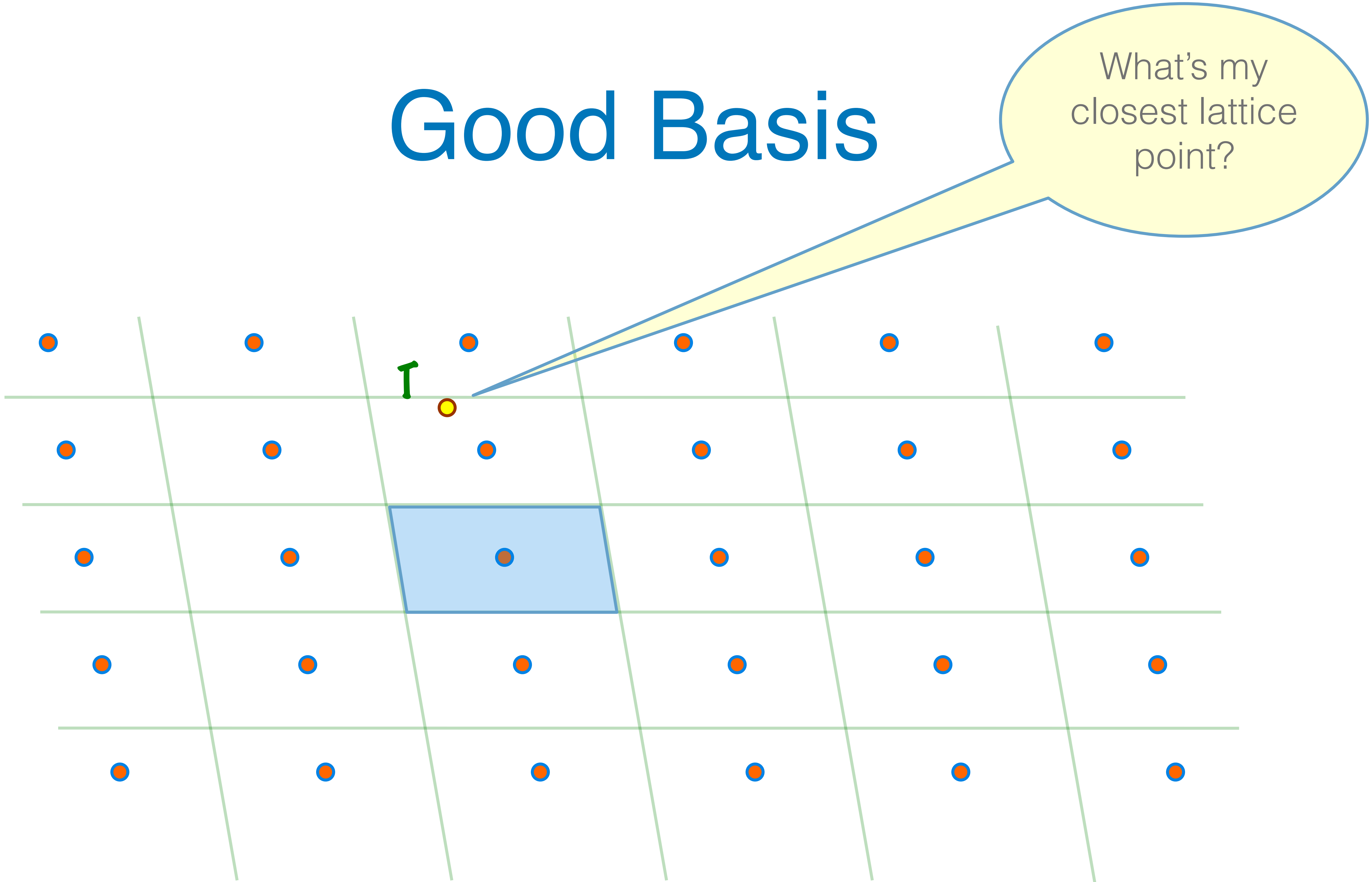


Good Basis



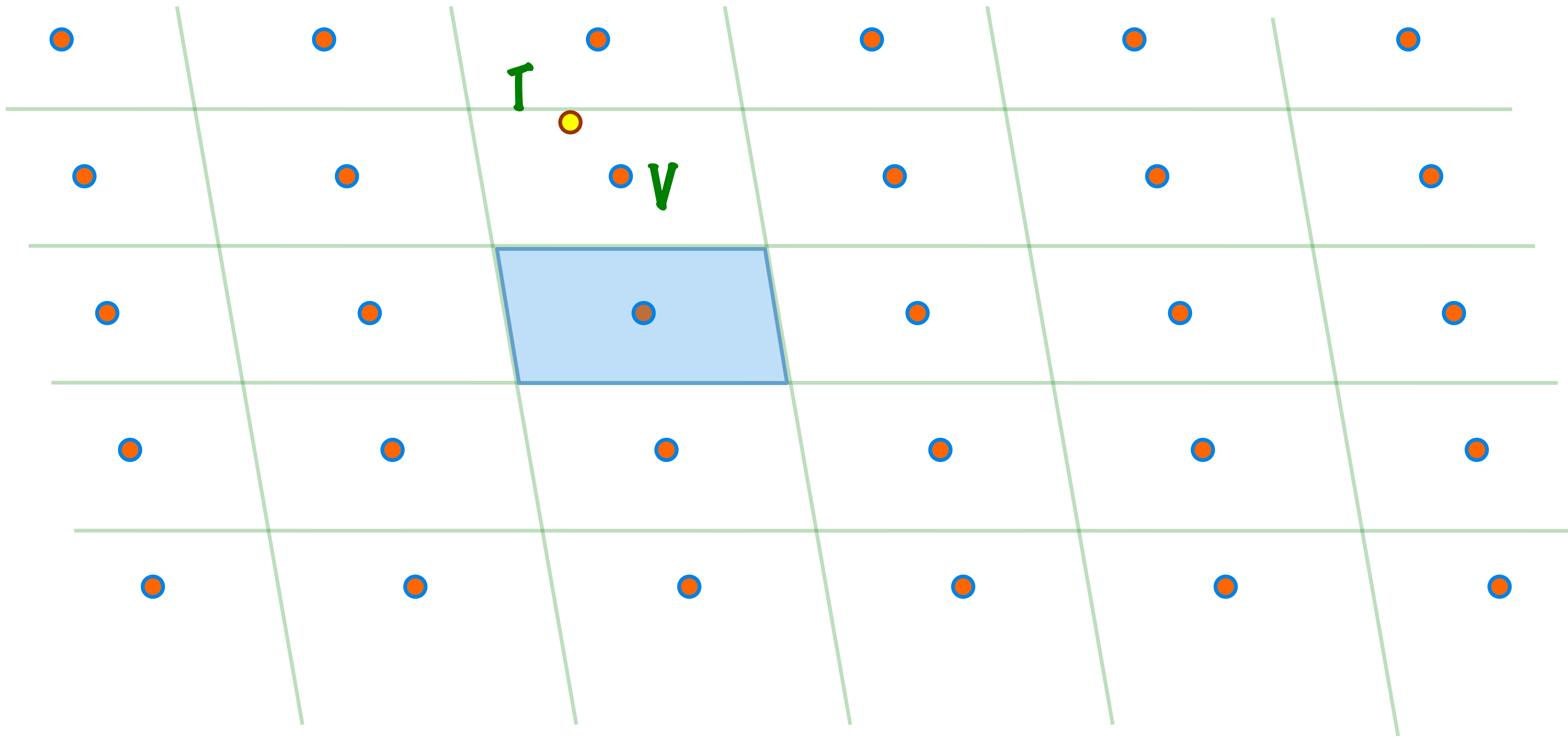
“Quite short” and “nearly orthogonal”

Good Basis

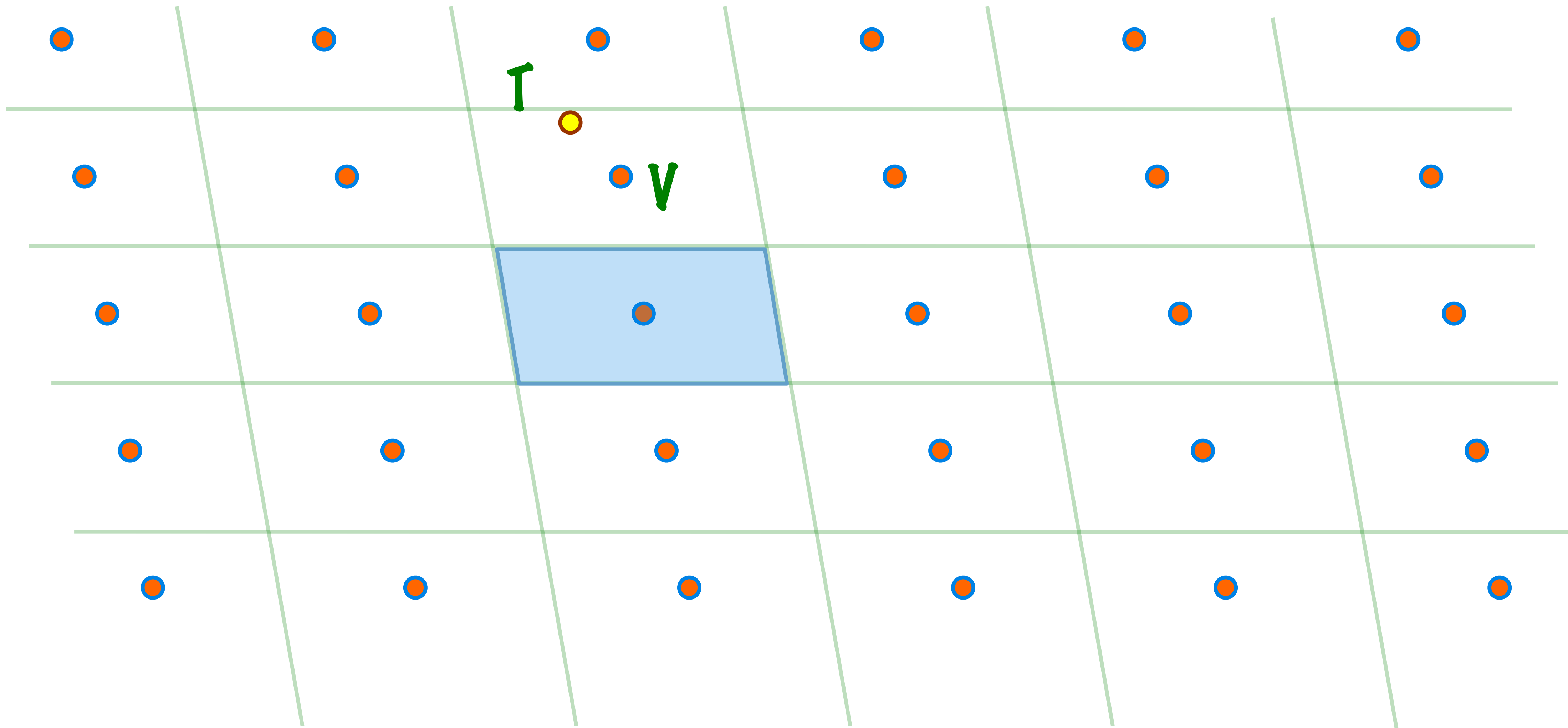


"Quite short" and "nearly orthogonal"

Good Basis

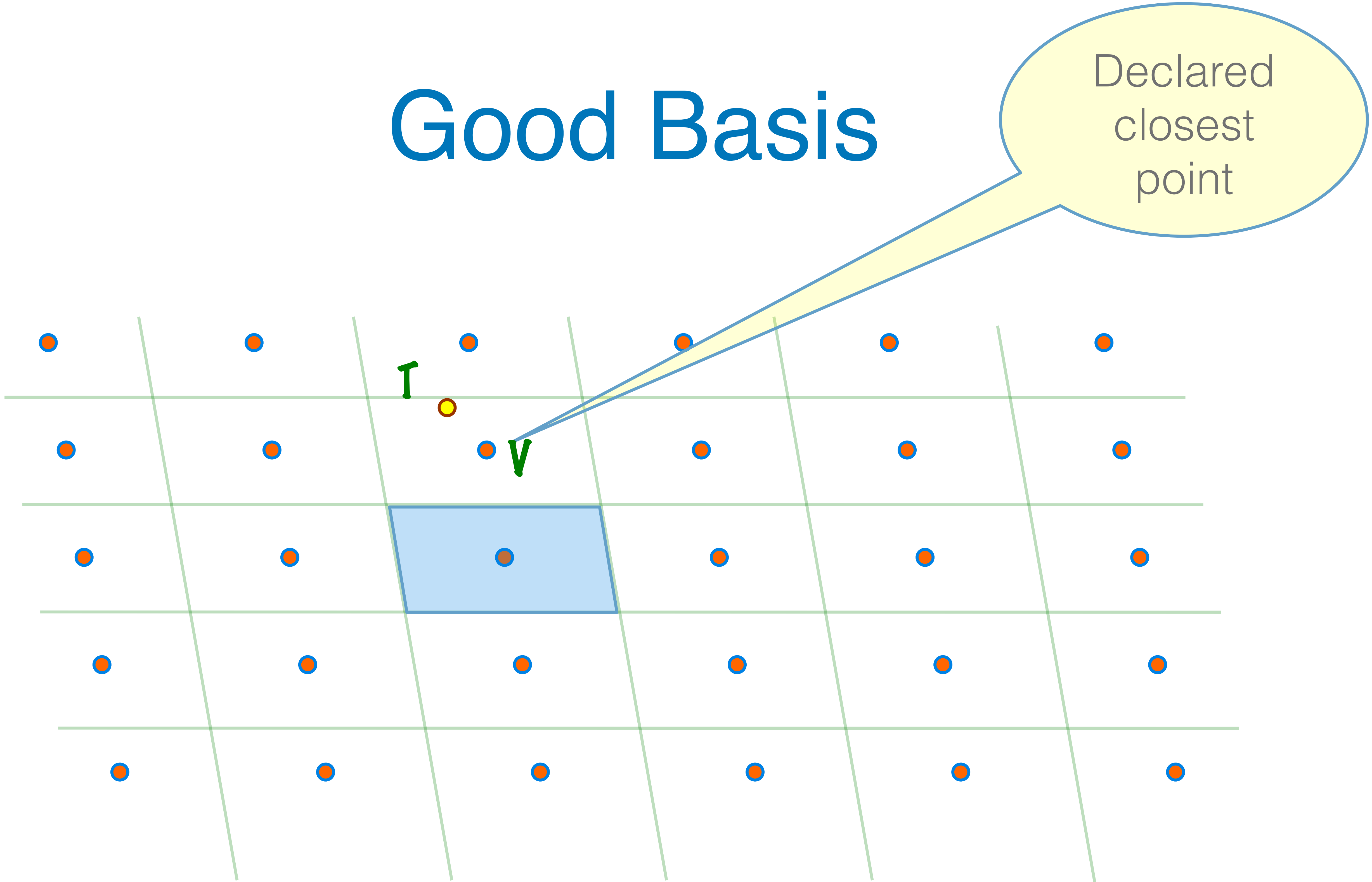


Good Basis



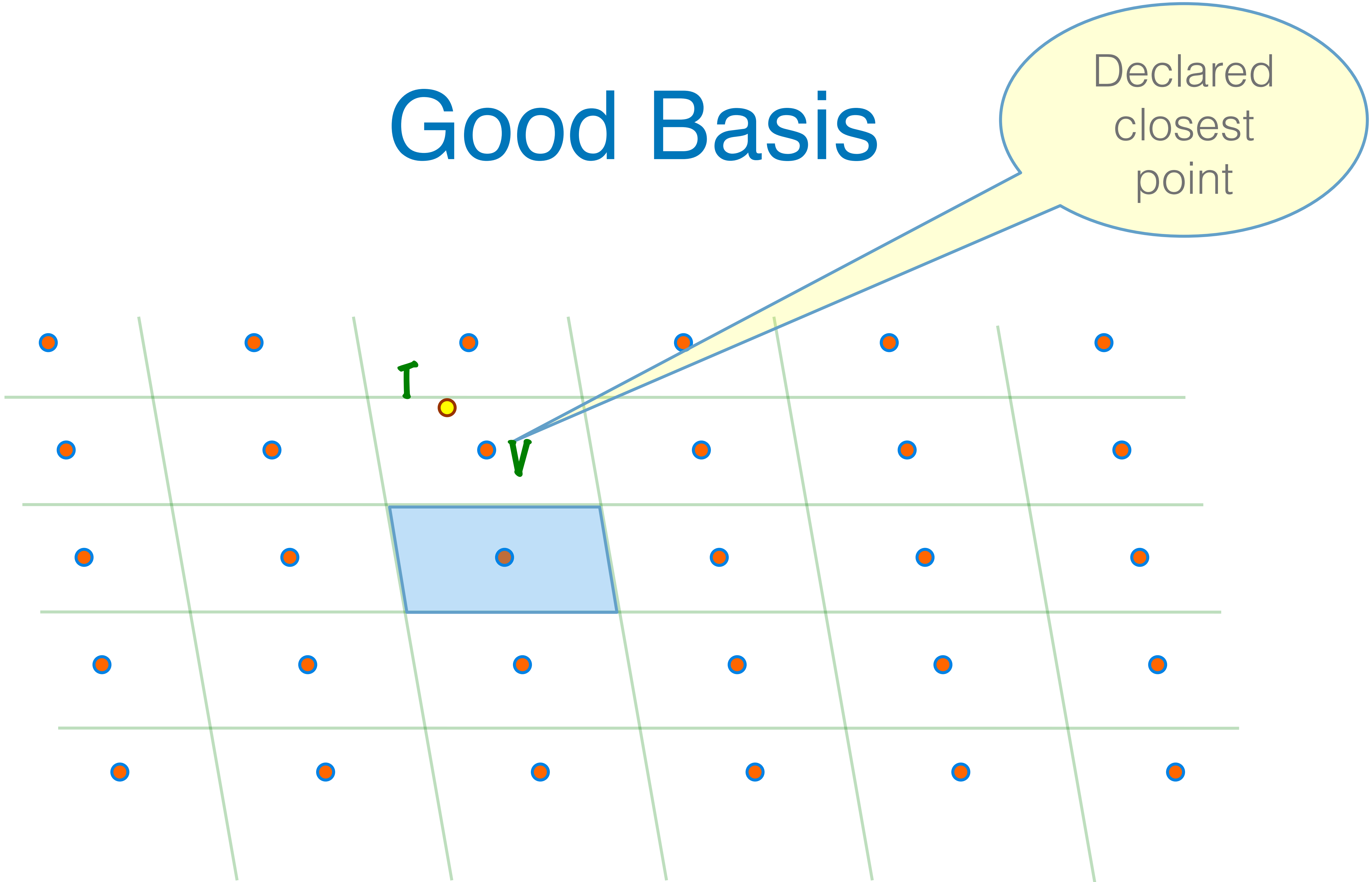
Output center of parallelopipid containing T

Good Basis



Output center of parallelopipid containing T

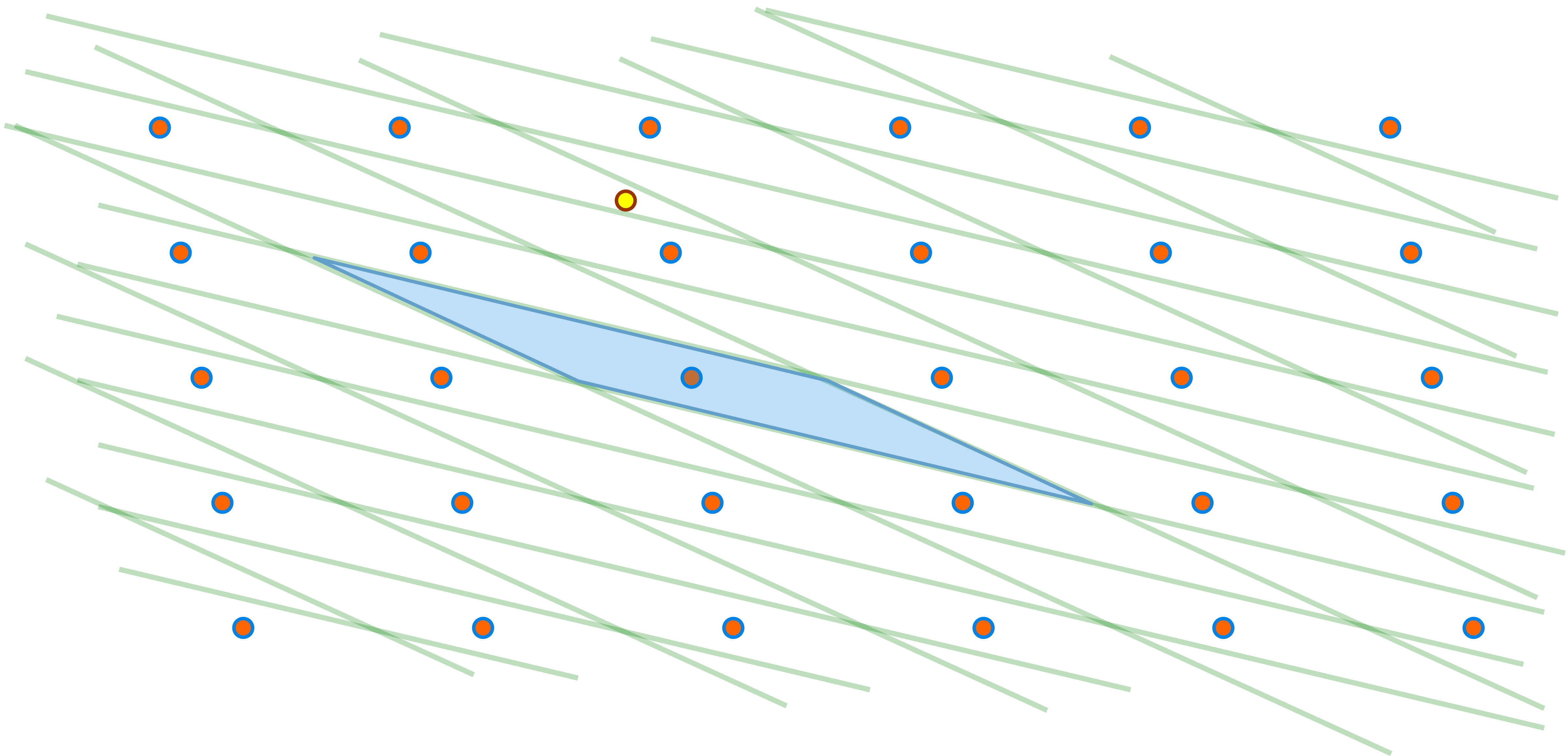
Good Basis



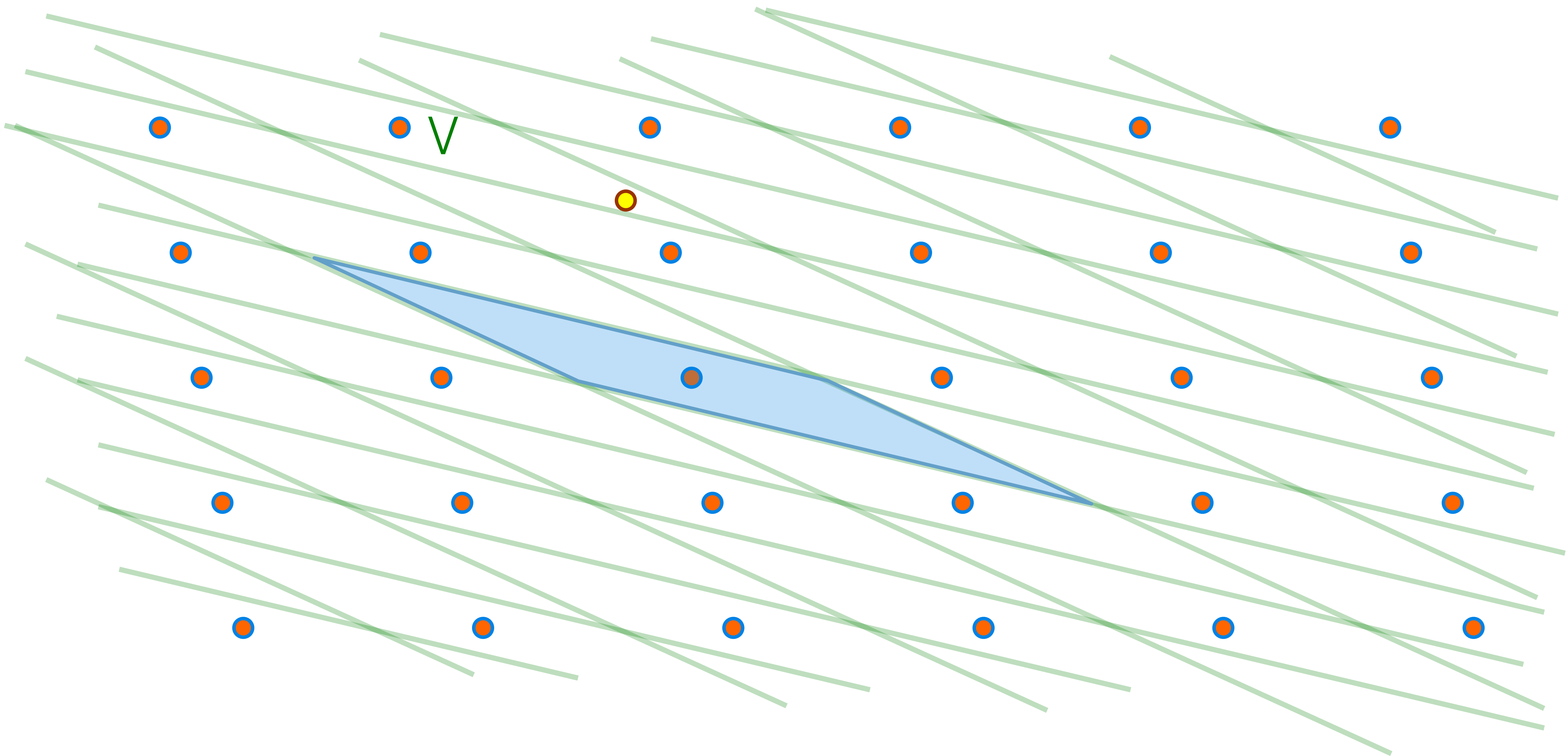
Output center of parallelopipid containing T

Pretty Accurate…

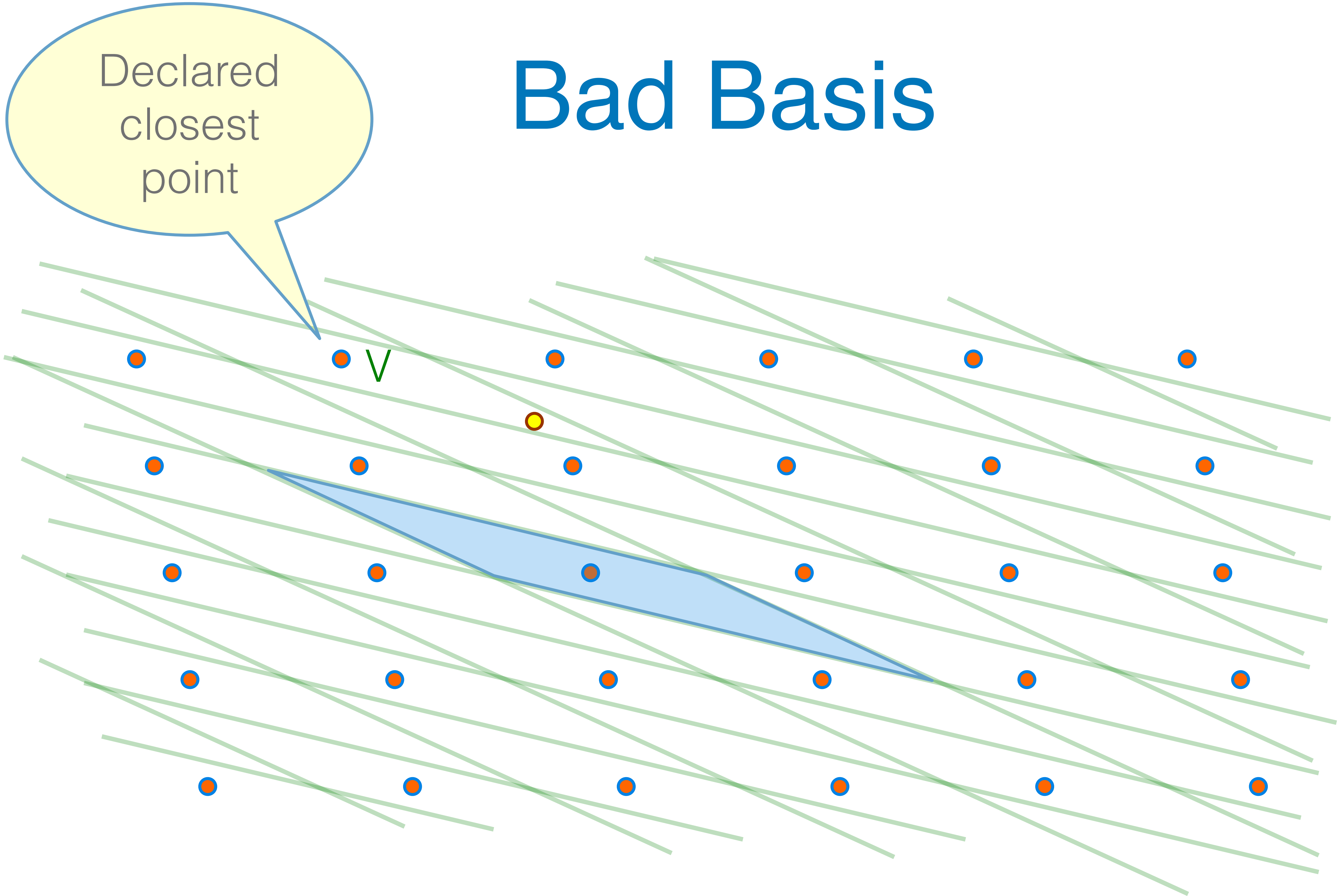
Bad Basis



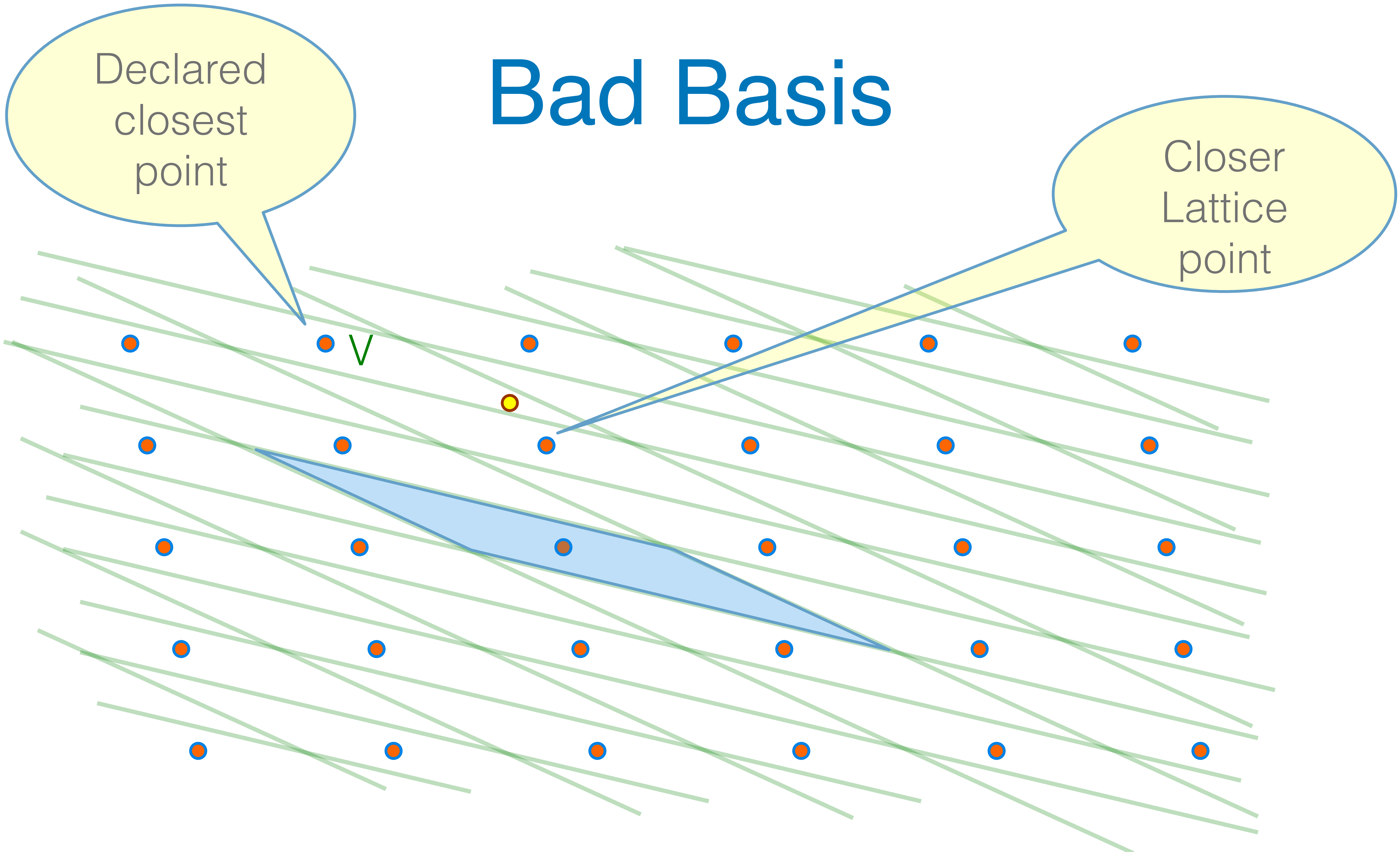
Bad Basis



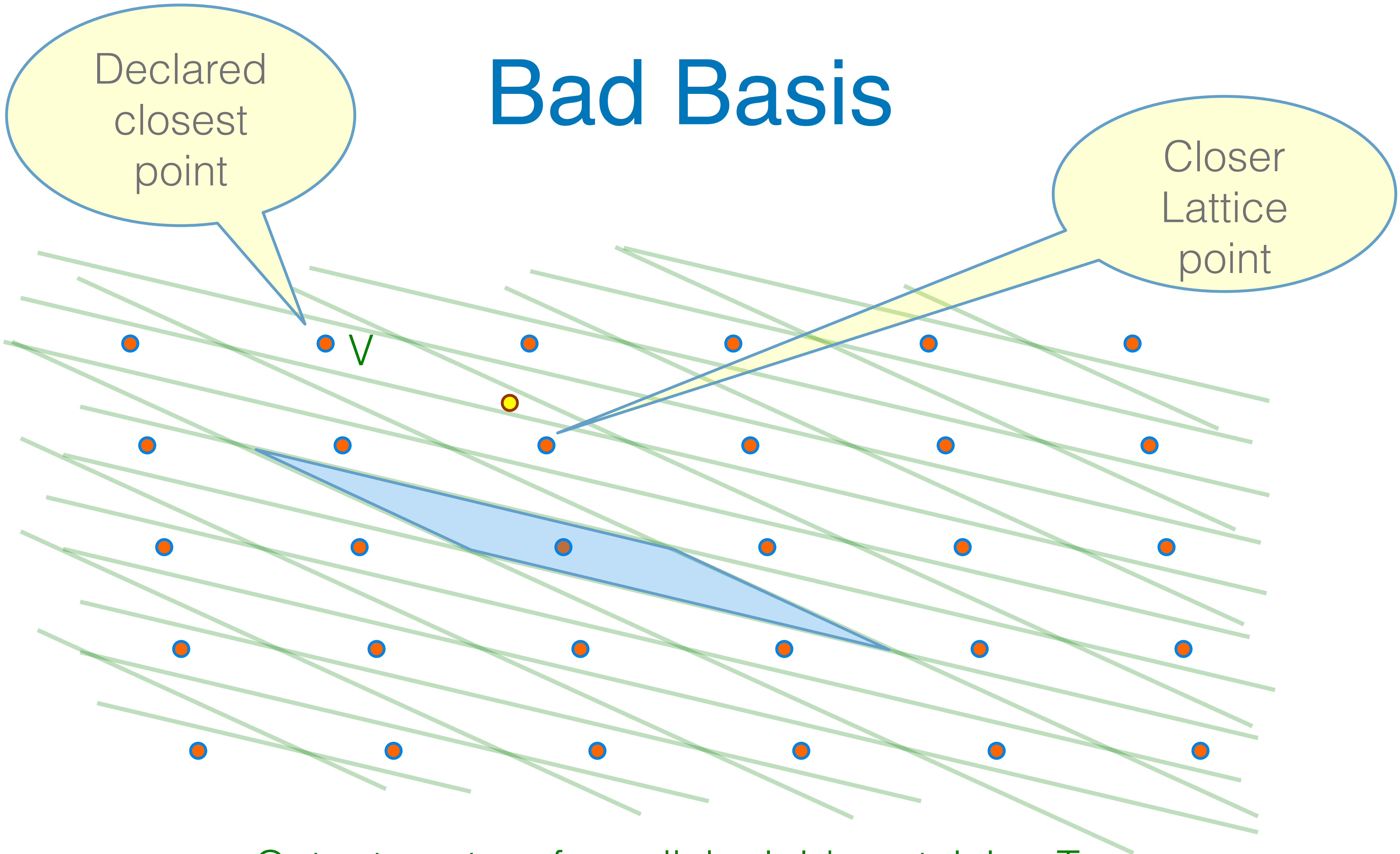
Bad Basis



Bad Basis

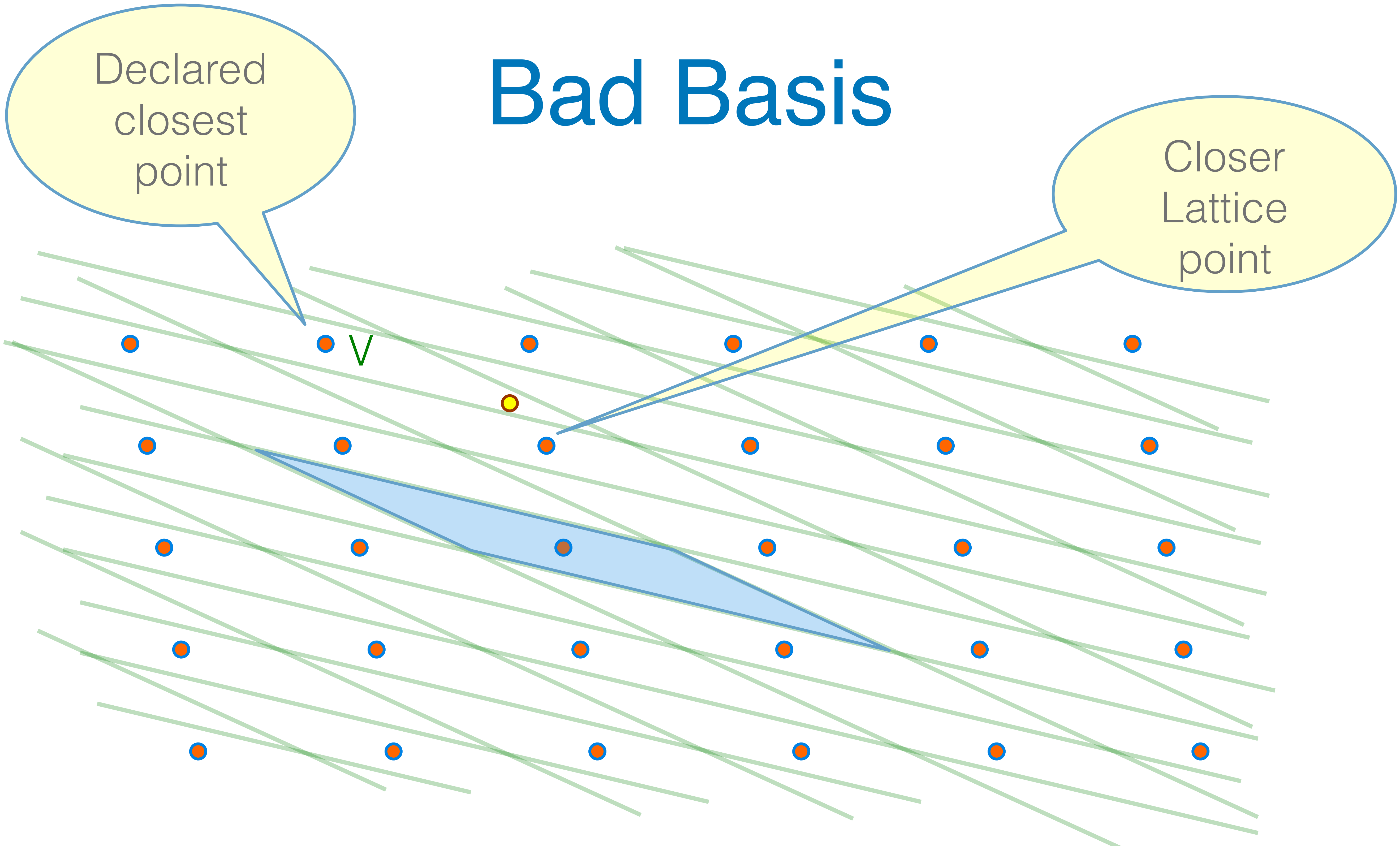


Bad Basis



Output center of parallelopipid containing T

Bad Basis



Output center of parallelopipid containing T

Not So Accurate…

Basis quality and Hardness

- SVP, CVP, SIS (...) hard given arbitrary (bad) basis
- Some hard lattice problems are easy given a good basis
- Will exploit this asymmetry

Basis quality and Hardness

- SVP, CVP, SIS (...) hard given arbitrary (bad) basis
- Some hard lattice problems are easy given a good basis
- Will exploit this asymmetry

Use Short Basis as Cryptographic Trapdoor!

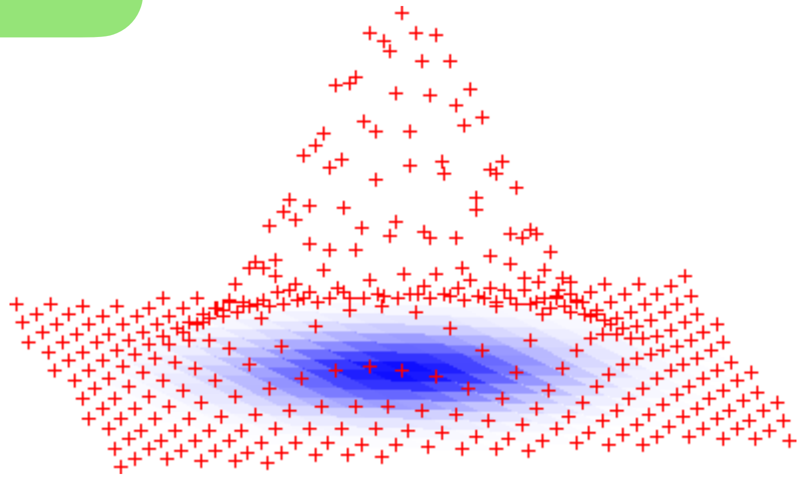
Lattice Trapdoors (Type 1)

Lattice Trapdoors (Type 1)

Inverting Our Function

Lattice Trapdoors (Type 1)

Inverting Our Function



Recall $\mathbf{u} = f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\ \mathbf{x} \mod q$

Want

$$\mathbf{x}' \leftarrow = f_{\mathbf{A}}^{-1}(\mathbf{u})$$

with prob $\propto \exp(-\|\mathbf{x}'\|^2/\sigma^2)$

Lattice Trapdoors (Type 1)

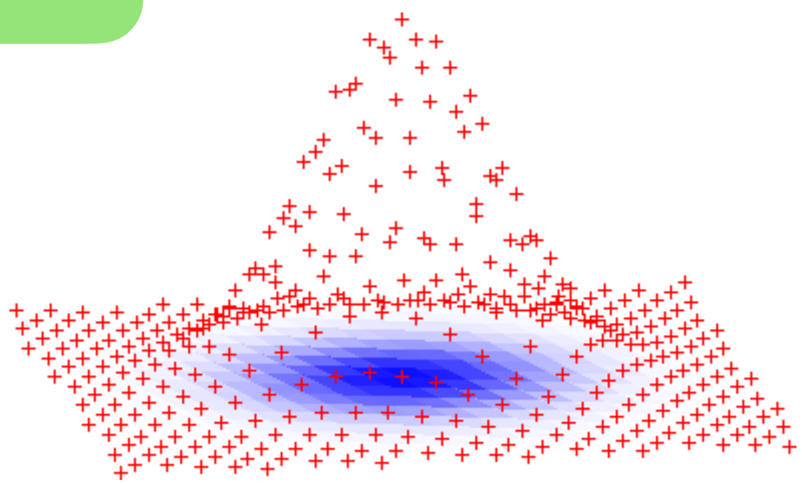
Inverting Our Function

Recall $\mathbf{u} = f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\ \mathbf{x} \mod q$

Want

$$\mathbf{x}' \leftarrow = f_{\mathbf{A}}^{-1}(\mathbf{u})$$

with prob $\propto \exp(-\|\mathbf{x}'\|^2/\sigma^2)$



The Lattice

Lattice Trapdoors (Type 1)

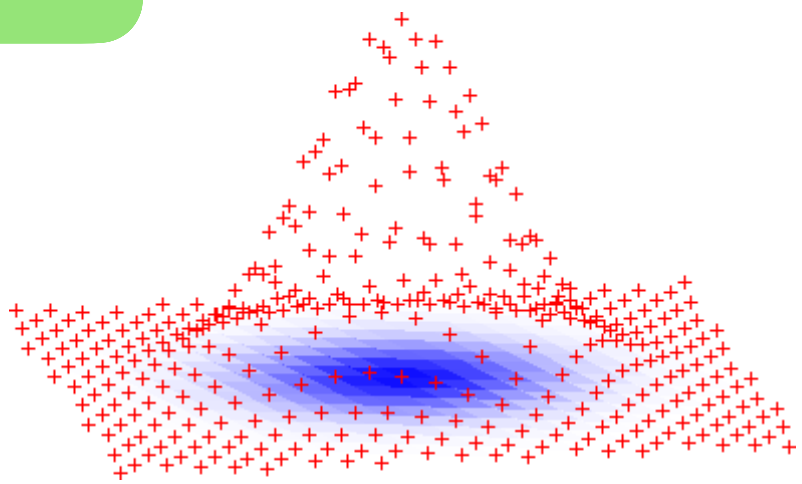
Inverting Our Function

Recall $\mathbf{u} = f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\ \mathbf{x} \mod q$

Want

$$\mathbf{x}' \leftarrow = f_{\mathbf{A}}^{-1}(\mathbf{u})$$

with prob $\propto \exp(-\|\mathbf{x}'\|^2/\sigma^2)$



The Lattice

$$\mathbf{\Lambda} = \{\mathbf{x} : \mathbf{A}\mathbf{x} = 0 \mod q\} \subseteq \mathbb{Z}_q^m$$

Lattice Trapdoors (Type 1)

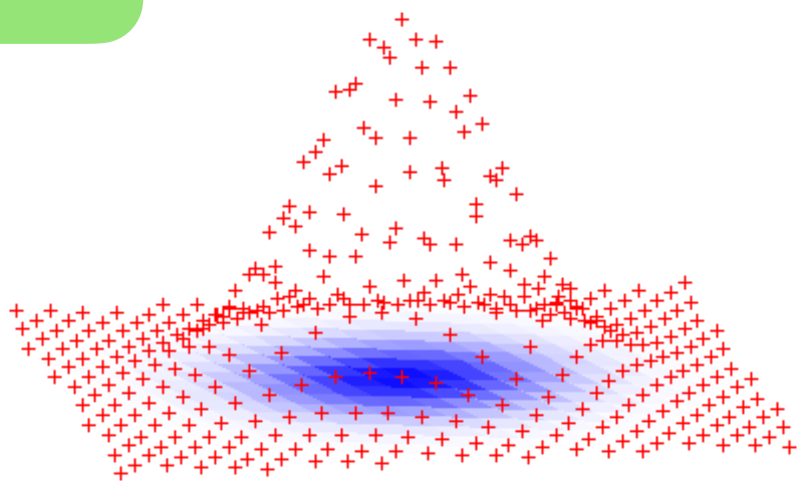
Inverting Our Function

Recall $\mathbf{u} = f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\ \mathbf{x} \mod q$

Want

$$\mathbf{x}' \leftarrow = f_{\mathbf{A}}^{-1}(\mathbf{u})$$

with prob $\propto \exp(-\|\mathbf{x}'\|^2/\sigma^2)$



The Lattice

$$\boldsymbol{\Lambda} = \{\mathbf{x} : \mathbf{A}\mathbf{x} = 0 \mod q\} \subseteq \mathbb{Z}_q^m$$

Short basis for $\boldsymbol{\Lambda}$ lets us sample from $f_{\mathbf{A}}^{-1}(\mathbf{u})$ with correct distribution!

Two Questions

Two Questions

1. How to get short basis

Two Questions

1. How to get short basis

Two Questions

1. How to get short basis

2. How to use short basis

Lattice Trapdoors (Type 2)

Lattice Trapdoors (Type 2)

Lattice Trapdoors (Type 2)

Not a short basis but

Lattice Trapdoors (Type 2)

Not a short basis but

- Just as powerful

Lattice Trapdoors (Type 2)

Not a short basis but

- Just as powerful
- More efficient

Lattice Trapdoors (Type 2)

Not a short basis but

- Just as powerful
- More efficient
- Better parameters

Lattice Trapdoors (Type 2)

Not a short basis but

- Just as powerful
- More efficient
- Better parameters
- Implies Type 1 trapdoors

Type 2 Trapdoors [MP12]

Recall $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\ \mathbf{x} \mod q \in \mathbb{Z}_q^n$ and $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t \mathbf{A} + \mathbf{e}^t \mod q \in \mathbb{Z}_q^m$

Type 2 Trapdoors [MP12]

Recall $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\ \mathbf{x} \mod q \in \mathbb{Z}_q^n$ and $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \mod q \in \mathbb{Z}_q^m$

Design $f_{\mathbf{G}}^{-1},\ g_{\mathbf{G}}^{-1}$
for Gadget Matrix G
(fixed, public, offline)

1

Type 2 Trapdoors [MP12]

Recall $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\ \mathbf{x} \mod q \in \mathbb{Z}_q^n$ and $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \mod q \in \mathbb{Z}_q^m$

Design $f_{\mathbf{G}}^{-1},\ g_{\mathbf{G}}^{-1}$ for Gadget Matrix G (fixed, public, offline)

Randomize G ↔ A via nice unimodular transformation

1

2

Type 2 Trapdoors [MP12]

Recall $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\ \mathbf{x} \mod q \in \mathbb{Z}_q^n$ and $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \mod q \in \mathbb{Z}_q^m$

Design $f_{\mathbf{G}}^{-1}$, $g_{\mathbf{G}}^{-1}$
for Gadget Matrix G
(fixed, public, offline)

Randomize G $\leftrightarrow$ A via
<u>nice</u> unimodular
transformation

Reduce
$f_{\mathbf{A}}^{-1}$, $g_{\mathbf{A}}^{-1}$
to
$f_{\mathbf{G}}^{-1}$, $g_{\mathbf{G}}^{-1}$

1

2

3

Type 2 Trapdoors [MP12]

Recall $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\ \mathbf{x} \mod q \in \mathbb{Z}_q^n$ and $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \mod q \in \mathbb{Z}_q^m$

Design $f_{\mathbf{G}}^{-1},\ g_{\mathbf{G}}^{-1}$
for Gadget Matrix G
(fixed, public, offline)

Randomize G ↔ A via
nice unimodular
transformation

Reduce
$f_{\mathbf{A}}^{-1},\ g_{\mathbf{A}}^{-1}$
to
$f_{\mathbf{G}}^{-1},\ g_{\mathbf{G}}^{-1}$

1

2

3

Transformation in Step 2 is the trapdoor!

Step 1: $f_{\mathbf{G}}^{-1},\ g_{\mathbf{G}}^{-1}$ for Gadget G

Recall $f_{\mathbf{G}}(\mathbf{x}) = \mathbf{G}\ \mathbf{x} \mod q \in \mathbb{Z}_q^n$ and $g_{\mathbf{G}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{G} + \mathbf{e}^t \mod q \in \mathbb{Z}_q^m$

Let $q = 2^k$ and $\mathbf{g} = [1, 2, 4, \cdots, 2^{k-1}] \in \mathbb{Z}_q^{1\times k}$

Invert LWE: find $s \in Z_q$ **s.t.** $s \cdot \mathbf{g} + \mathbf{e} = [s + e_0, 2s + e_1, \ \cdots\ 2^{k-1}s + e_{k-1}]$

Step 1: $f_{\mathbf{G}}^{-1}, \; g_{\mathbf{G}}^{-1}$ for Gadget G

Recall $f_{\mathbf{G}}(\mathbf{x}) = \mathbf{G}\,\mathbf{x} \mod q \in \mathbb{Z}_q^n$ and $g_{\mathbf{G}}(\mathbf{s},\mathbf{e}) = \mathbf{s}^t\mathbf{G} + \mathbf{e}^t \mod q \in \mathbb{Z}_q^m$

Let $q = 2^k$ and $\mathbf{g} = [1, 2, 4, \cdots, 2^{k-1}] \in \mathbb{Z}_q^{1\times k}$

Invert LWE: find $s \in Z_q$ **s.t.** $s \cdot \mathbf{g} + \mathbf{e} = [s + e_0, 2s + e_1, \; \cdots \; 2^{k-1}s + e_{k-1}]$

- Get lsb(s) from $2^{k-1}s + e_{k-1}$
- Then get next bit of s and so on.
- Works as long as every $e_i \in [-q/4, q/4)$

Step 1: $f_{\mathbf{G}}^{-1}, \ g_{\mathbf{G}}^{-1}$ for Gadget G

Recall $f_{\mathbf{G}}(\mathbf{x}) = \mathbf{G}\,\mathbf{x} \mod q \in \mathbb{Z}_q^n$ and $g_{\mathbf{G}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{G} + \mathbf{e}^t \mod q \in \mathbb{Z}_q^m$

Let $q = 2^k$ and $\mathbf{g} = [1, 2, 4, \cdots, 2^{k-1}] \in \mathbb{Z}_q^{1\times k}$

Invert LWE: find $s \in Z_q$ **s.t.** $s \cdot \mathbf{g} + \mathbf{e} = [s + e_0, 2s + e_1, \ \cdots \ 2^{k-1}s + e_{k-1}]$

- Get lsb(s) from $2^{k-1}s + e_{k-1}$
- Then get next bit of s and so on.
- Works as long as every $e_i \in [-q/4, q/4)$

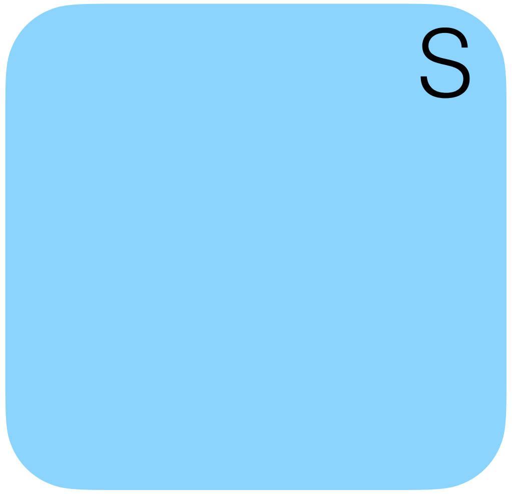
Gaussian from shifted lattice 2Z + u

Invert SIS: sample Gaussian preimage **x** s.t. $u = \langle \mathbf{g}\ \mathbf{x} \rangle \mod q$

- For $i \in [0, \ldots, k-1]$, choose $x_i \leftarrow (2\mathbb{Z} + u)$, $u \leftarrow (u - x_i)/2 \in \mathbb{Z}$
- Let k= 2. $x_0 \leftarrow (2z_0 + u)$, $u \leftarrow (u - 2z_0 - u)/2 = -z_0$

$$x_1 \leftarrow (2z_1 - z_0)$$
$$\langle \mathbf{g}, \mathbf{x} \rangle = 2z_0 + u + 2(2z_1 - z_0) = u + 4z_1 = u \mod 4$$

Step 1: $f_{\mathbf{G}}^{-1}$, $g_{\mathbf{G}}^{-1}$ for Gadget G

Want $\mathbf{g} = [1, 2, 4, \cdots, 2^{k-1}]$ 

$=$ 0 mod q

Step 1: $f_{\mathbf{G}}^{-1},\ g_{\mathbf{G}}^{-1}$ for Gadget G

Note $\mathbf{g} = [1, 2, 4, \cdots, 2^{k-1}]$

$$\underbrace{\begin{bmatrix} 2 & & & & \\ -1 & 2 & & & \\ & -1 & 2 & & \\ & & \ddots & \ddots & \\ & & & 2 & \\ & & & -1 & 2 \end{bmatrix}}_{S} = 0 \bmod q$$

Step 1: $f_{\mathbf{G}}^{-1}$, $g_{\mathbf{G}}^{-1}$ for Gadget G

Note $\mathbf{g} = [1, 2, 4, \cdots, 2^{k-1}]$

$$S = \begin{bmatrix} 2 & & & & \\ -1 & 2 & & & \\ & -1 & 2 & & \\ & & \ddots & \ddots & \\ & & & 2 & \\ & & & -1 & 2 \end{bmatrix} = 0 \bmod q$$

S is Short Basis for $\mathbf{g} = [1, 2, 4, \cdots, 2^{k-1}]$

Step 1: $f_{\mathbf{G}}^{-1},\ g_{\mathbf{G}}^{-1}$ for Gadget G

Note $\mathbf{g} = [1, 2, 4, \cdots, 2^{k-1}]$

$$S = \begin{bmatrix} 2 & & & & & \\ -1 & 2 & & & & \\ & -1 & 2 & & & \\ & & & \ddots & & \\ & & & & 2 & \\ & & & & -1 & 2 \end{bmatrix} \quad = \quad 0 \bmod q$$

S is Short Basis for $\mathbf{g} = [1, 2, 4, \cdots, 2^{k-1}]$

Define gadget G : $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g}$

$$\begin{bmatrix} \cdots\ \mathbf{g}\ \cdots & & & \\ & \cdots\ \mathbf{g}\ \cdots & & \\ & & \ddots & \\ & & & \cdots\ \mathbf{g}\ \cdots \end{bmatrix} \in \mathbb{Z}_q^{n \times nk}$$

Step 1: $f_{\mathbf{G}}^{-1},\ g_{\mathbf{G}}^{-1}$ for Gadget G

Note $\mathbf{g} = [1, 2, 4, \cdots, 2^{k-1}]$

S

$$\begin{bmatrix} 2 & & & & \\ -1 & 2 & & & \\ & -1 & 2 & & \\ & & \ddots & & \\ & & & 2 & \\ & & & -1 & 2 \end{bmatrix} \quad = \quad 0 \bmod q$$

S is Short Basis for $\mathbf{g} = [1, 2, 4, \cdots, 2^{k-1}]$

Define gadget G : $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g}$

$$\begin{bmatrix} \cdots\ \mathbf{g}\ \cdots & & & \\ & \cdots\ \mathbf{g}\ \cdots & & \\ & & \ddots & \\ & & & \cdots\ \mathbf{g}\ \cdots \end{bmatrix} \in \mathbb{Z}_q^{n \times nk}$$

$f_{\mathbf{G}}^{-1},\ g_{\mathbf{G}}^{-1}$ reduce to n parallel, offline calls to $f_{\mathbf{g}}^{-1},\ g_{\mathbf{g}}^{-1}$

Step 2: Randomize G to A

Step 2: Randomize G to A

1. Sample $\mathbf{B} \in \mathbb{Z}_q^{n \times m'}$, short Gaussian $\mathbf{R} \in \mathbb{Z}_q^{m' \times n \log q}$,

Step 2: Randomize G to A

1. Sample $\mathbf{B} \in \mathbb{Z}_q^{n\times m'}$, short Gaussian $\mathbf{R} \in \mathbb{Z}_q^{m'\times n \log q}$,

2. Define $\mathbf{A} = \begin{bmatrix} \mathrm{B} & \mathrm{G} \end{bmatrix} \left[\begin{array}{c|c} \mathrm{I} & \mathrm{-R} \\ \hline & \mathrm{I} \end{array}\right]$

Step 2: Randomize G to A

1. Sample $\mathbf{B} \in \mathbb{Z}_q^{n \times m'}$, short Gaussian $\mathbf{R} \in \mathbb{Z}_q^{m' \times n \log q}$,

2. Define $\mathbf{A} = \begin{bmatrix} \mathrm{B} & \mathrm{G} \end{bmatrix} \begin{bmatrix} \mathrm{I} & \mathrm{-R} \\ & \mathrm{I} \end{bmatrix}$

$= \begin{bmatrix} \mathrm{B} & \mathrm{G - BR} \end{bmatrix}$

Step 2: Randomize G to A

1. Sample $\mathbf{B} \in \mathbb{Z}_q^{n \times m'}$, short Gaussian $\mathbf{R} \in \mathbb{Z}_q^{m' \times n \log q}$,

2. Define $\mathbf{A} = \begin{bmatrix} B & G \end{bmatrix} \begin{bmatrix} I & -R \\ & I \end{bmatrix}$

$$= \left[\begin{array}{c|c} B & G - BR \end{array}\right]$$

A is uniform by leftover hash lemma!

Leftover Hash Lemma (oversimplified)

Leftover Hash Lemma (oversimplified)

Let $\mathbf{B} \in \mathbb{Z}_q^{n \times m'}$ uniform & $\mathbf{R} \in \mathbb{Z}_q^{m' \times n \log q}$ Gaussian

If $m' \approx n \log q$, then,

$$(\mathbf{B}, \mathbf{BR}) \approx (\mathbf{B}, \mathbf{U})$$

Leftover Hash Lemma (oversimplified)

Let $\mathbf{B} \in \mathbb{Z}_q^{n \times m'}$ uniform & $\mathbf{R} \in \mathbb{Z}_q^{m' \times n \log q}$ Gaussian

If $m' \approx n \log q$, then,

$$(\mathbf{B}, \mathbf{BR}) \approx (\mathbf{B}, \mathbf{U})$$

Hence **A** = $\left[\, \mathrm{B} \;\middle|\; \mathrm{G} - \mathrm{BR} \,\right]$ uniform

Step 2: Randomize G to A

Step 2: Randomize G to A

Have $\mathbf{A} =$

B	G - BR

Step 2: Randomize G to A

Have $\mathbf{A} =$

B	G - BR

Define: **R** is a trapdoor for **A** with tag $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$,

If $$\mathbf{A} \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{H} \cdot \mathbf{G}$$

Step 2: Randomize G to A

Have $\mathbf{A} =$

B	G - BR

Define: **R** is a trapdoor for **A** with tag $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$,

If $\mathbf{A} \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{H} \cdot \mathbf{G}$

Basis S **for** $\Lambda^{\perp}(\mathbf{G})$

&

Trapdoor R for A

→

Step 2: Randomize G to A

Have $\mathbf{A} = \left[\; \mathbf{B} \;\middle|\; \mathbf{G} - \mathbf{BR} \;\right]$

Define: **R** is a trapdoor for **A** with tag $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$,

If $$\mathbf{A} \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{H} \cdot \mathbf{G}$$

Basis $\mathbf{S}$ **for** $\Lambda^{\perp}(\mathbf{G})$ & **Trapdoor R for A** $\Rightarrow$ **Basis** $\mathbf{S_A}$ **for** $\Lambda^{\perp}(\mathbf{A})$

Step 3: Reduce $f_{\mathbf{A}}^{-1}, g_{\mathbf{A}}^{-1}$ to $f_{\mathbf{G}}^{-1}, g_{\mathbf{G}}^{-1}$

Suppose **R** is a trapdoor for **A** with tag $\mathbf{I} \in \mathbb{Z}_q^{n \times n}$,

$$\mathbf{A} \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{G}$$

Step 3: Reduce $f_{\mathbf{A}}^{-1}, g_{\mathbf{A}}^{-1}$ to $f_{\mathbf{G}}^{-1}, g_{\mathbf{G}}^{-1}$

Suppose **R** is a trapdoor for **A** with tag $\mathbf{I} \in \mathbb{Z}_q^{n \times n}$,

$$\mathbf{A} \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{G}$$

Inverting LWE

Step 3: Reduce $f_{\mathbf{A}}^{-1}, \ g_{\mathbf{A}}^{-1}$ to $f_{\mathbf{G}}^{-1}, \ g_{\mathbf{G}}^{-1}$

Suppose **R** is a trapdoor for **A** with tag $\mathbf{I} \in \mathbb{Z}_q^{n \times n}$,

$$\mathbf{A} \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{G}$$

Inverting LWE

Want:

- Given $\mathbf{b}^t = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \mod q$
- Find unique $(\mathbf{s}, \mathbf{e})$

Step 3: Reduce $f_{\mathbf{A}}^{-1}, g_{\mathbf{A}}^{-1}$ to $f_{\mathbf{G}}^{-1}, g_{\mathbf{G}}^{-1}$

Suppose **R** is a trapdoor for **A** with tag $\mathbf{I} \in \mathbb{Z}_q^{n\times n}$,

$$\mathbf{A} \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{G}$$

Inverting LWE

Want:

- Given $\mathbf{b}^t = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \mod q$
- Find unique $(\mathbf{s}, \mathbf{e})$

Compute:

$$\mathbf{b}^t \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{s}^t \cdot \mathbf{G} + \mathbf{e}^t \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \mod q$$

Works if $\mathbf{e}^t \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \in [-q/4, q/4)$

Step 3: Reduce $f_{\mathbf{A}}^{-1},\ g_{\mathbf{A}}^{-1}$ to $f_{\mathbf{G}}^{-1},\ g_{\mathbf{G}}^{-1}$

Inverting SIS

$$\mathbf{A} \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{G}$$

Step 3: Reduce $f_{\mathbf{A}}^{-1},\ g_{\mathbf{A}}^{-1}$ to $f_{\mathbf{G}}^{-1},\ g_{\mathbf{G}}^{-1}$

Inverting SIS

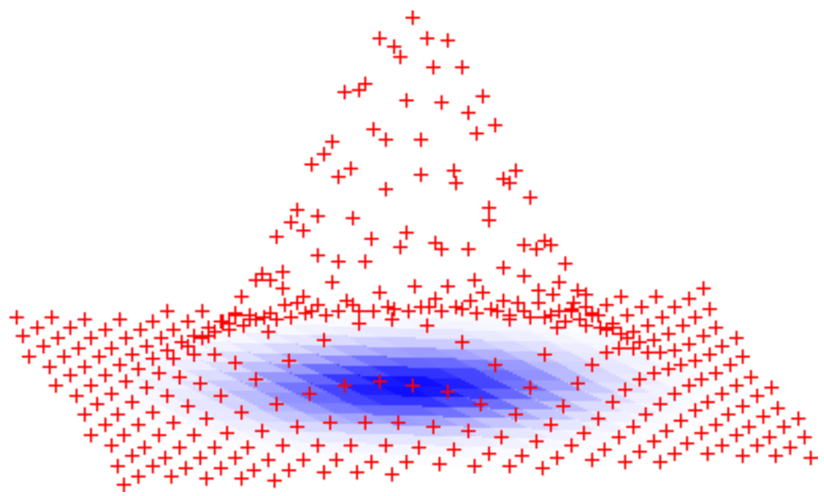
$$\mathbf{A} \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{G}$$

Want:

- Given $\mathbf{u} = f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\ \mathbf{x} \mod q$
- Sample

$$\mathbf{x}' \leftarrow = f_{\mathbf{A}}^{-1}(\mathbf{u})$$

with prob $\propto \exp(-\|\mathbf{x}'\|^2/\sigma^2)$



Step 3: Reduce $f_{\mathbf{A}}^{-1}, g_{\mathbf{A}}^{-1}$ to $f_{\mathbf{G}}^{-1}, g_{\mathbf{G}}^{-1}$

Inverting SIS

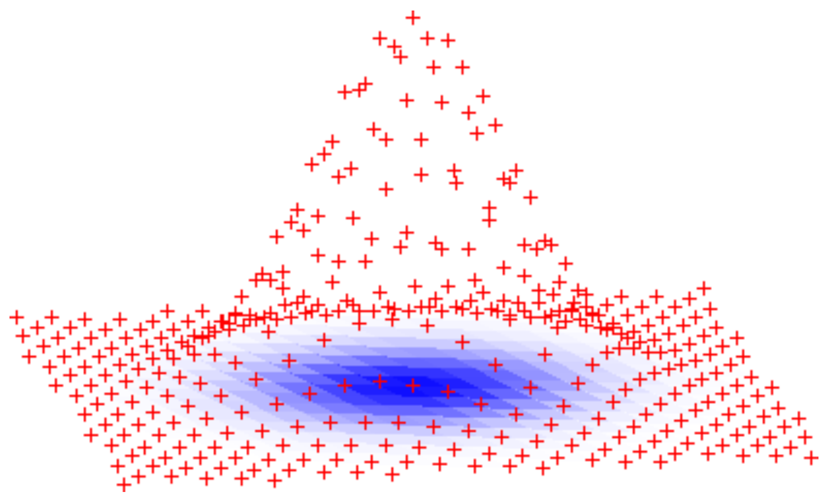
$$\mathbf{A} \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{G}$$

Want:

- Given $\mathbf{u} = f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\ \mathbf{x} \mod q$
- Sample

$$\mathbf{x}' \leftarrow = f_{\mathbf{A}}^{-1}(\mathbf{u})$$

with prob $\propto \exp(-\|\mathbf{x}'\|^2/\sigma^2)$



Compute:

Sample $\mathbf{z} \leftarrow f_{\mathbf{G}}^{-1}(\mathbf{u})$

Output $\mathbf{x} = \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \cdot \mathbf{z}$

Then,

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{A} \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \cdot \mathbf{z} = \mathbf{G} \cdot \mathbf{z} = \mathbf{u}$$

Step 3: Reduce $f_{\mathbf{A}}^{-1},\ g_{\mathbf{A}}^{-1}$ to $f_{\mathbf{G}}^{-1},\ g_{\mathbf{G}}^{-1}$

Are we done?

$$\mathbf{A} \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{G}$$

Step 3: Reduce $f_{\mathbf{A}}^{-1},\ g_{\mathbf{A}}^{-1}$ to $f_{\mathbf{G}}^{-1},\ g_{\mathbf{G}}^{-1}$

Are we done?

$$\mathbf{A} \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{G}$$

Compute:

Sample $\mathbf{z} \leftarrow f_{\mathbf{G}}^{-1}(\mathbf{u})$

Output $\mathbf{x} = \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \cdot \mathbf{z}$

Then,

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{A} \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \cdot \mathbf{z} = \mathbf{G} \cdot \mathbf{z} = \mathbf{u}$$

Step 3: Reduce $f_{\mathbf{A}}^{-1},\ g_{\mathbf{A}}^{-1}$ to $f_{\mathbf{G}}^{-1},\ g_{\mathbf{G}}^{-1}$

Are we done?

$$\mathbf{A} \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{G}$$

Compute:

Sample $\mathbf{z} \leftarrow f_{\mathbf{G}}^{-1}(\mathbf{u})$

Output $\mathbf{x} = \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \cdot \mathbf{z}$

Then,

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{A} \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \cdot \mathbf{z} = \mathbf{G} \cdot \mathbf{z} = \mathbf{u}$$

Covariance of x leaks R!

Step 3: Reduce $f_{\mathbf{A}}^{-1},\ g_{\mathbf{A}}^{-1}$ to $f_{\mathbf{G}}^{-1},\ g_{\mathbf{G}}^{-1}$

Are we done?

$$\mathbf{A} \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{G}$$

Compute:

Sample $\mathbf{z} \leftarrow f_{\mathbf{G}}^{-1}(\mathbf{u})$

Output $\mathbf{x} = \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \cdot \mathbf{z}$

Then,

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{A} \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \cdot \mathbf{z} = \mathbf{G} \cdot \mathbf{z} = \mathbf{u}$$

Covariance of x leaks R!

$$\Sigma := \mathbb{E}_{\mathbf{x}}\left[\mathbf{x} \cdot \mathbf{x}^t\right] = \mathbb{E}_{\mathbf{z}}\left[\mathbf{R} \cdot \mathbf{z}\mathbf{z}^t \cdot \mathbf{R}^t\right] \approx s^2 \cdot \mathbf{R}\mathbf{R}^t.$$

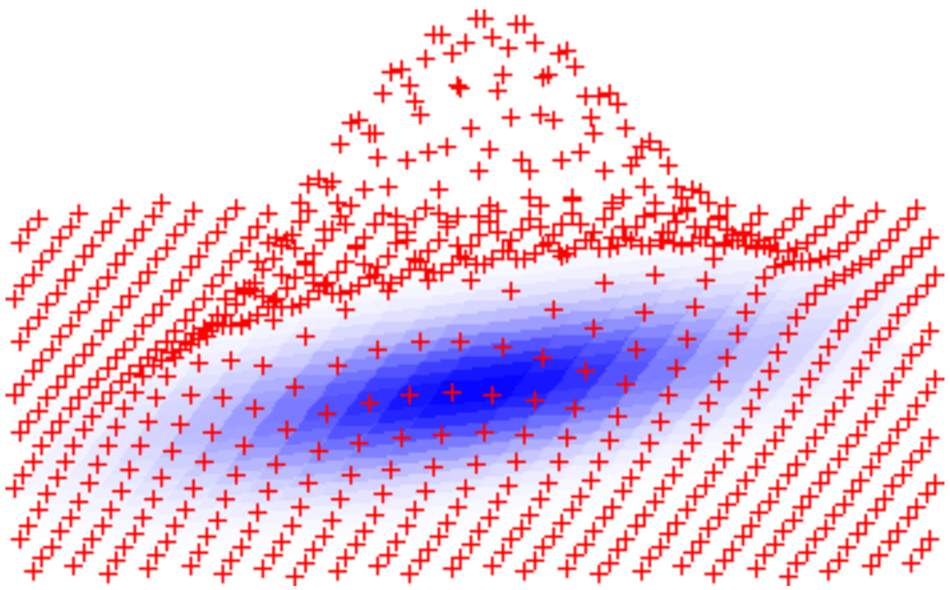


Image Credit: Chris Peikert

Step 3: Reduce $f_{\mathbf{A}}^{-1}$, $g_{\mathbf{A}}^{-1}$ to $f_{\mathbf{G}}^{-1}$, $g_{\mathbf{G}}^{-1}$

Step 3: Reduce $f_{\mathbf{A}}^{-1},\ g_{\mathbf{A}}^{-1}$ to $f_{\mathbf{G}}^{-1},\ g_{\mathbf{G}}^{-1}$

Want to output spherical Gaussian!
Covariance Matrix $s^2\mathbf{I}$

Step 3: Reduce $f_{\mathbf{A}}^{-1}, g_{\mathbf{A}}^{-1}$ to $f_{\mathbf{G}}^{-1}, g_{\mathbf{G}}^{-1}$

Want to output spherical Gaussian!
Covariance Matrix $s^2\mathbf{I}$

https://www.elegantthemes.com/

Fix using perturbation method [P'10]

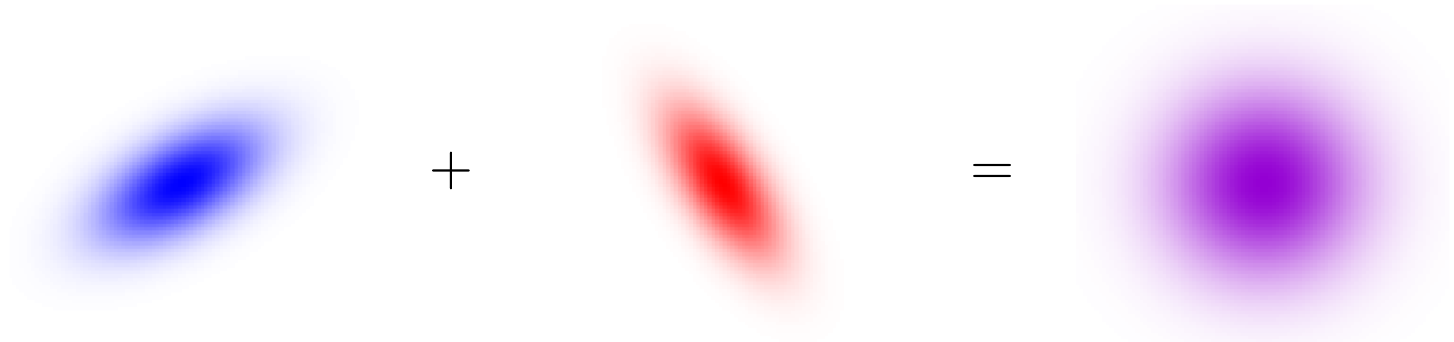
Step 3: Reduce $f_{\mathbf{A}}^{-1},\ g_{\mathbf{A}}^{-1}$ to $f_{\mathbf{G}}^{-1},\ g_{\mathbf{G}}^{-1}$

Want to output spherical Gaussian!
Covariance Matrix $s^2\mathbf{I}$

https://www.elegantthemes.com/

Fix using perturbation method [P'10]

Convolution of Gaussians



$$\mathbf{R}\mathbf{R}^t \quad + \quad (s^2\mathbf{I} - \mathbf{R}\mathbf{R}^t) \quad = \quad s^2\,\mathbf{I}$$

Step 3: Reduce $f_{\mathbf{A}}^{-1}, g_{\mathbf{A}}^{-1}$ to $f_{\mathbf{G}}^{-1}, g_{\mathbf{G}}^{-1}$

Want to output spherical Gaussian!
Covariance Matrix $s^2\mathbf{I}$

https://www.elegantthemes.com/

Fix using perturbation method [P'10]

Convolution of Gaussians

$$\mathbf{R}\mathbf{R}^t \quad + \quad (s^2\mathbf{I} - \mathbf{R}\mathbf{R}^t) \quad = \quad s^2\,\mathbf{I}$$

To fix covariance:

- Generate perturbation vector **p** with covariance $(s^2\mathbf{I} - \mathbf{R}\mathbf{R}^t)$
- Sample spherical **z** such that $\mathbf{G}\,\mathbf{z} = \mathbf{u} - \mathbf{A}\,\mathbf{p}$
- Output $\mathbf{x} = \mathbf{p} + \begin{bmatrix}\mathbf{R}\\ \mathbf{I}\end{bmatrix} \cdot \mathbf{z}$

Step 3: Reduce $f_{\mathbf{A}}^{-1}, g_{\mathbf{A}}^{-1}$ to $f_{\mathbf{G}}^{-1}, g_{\mathbf{G}}^{-1}$

Want to output spherical Gaussian!
Covariance Matrix $s^2\mathbf{I}$

https://www.elegantthemes.com/

Fix using perturbation method [P'10]

Convolution of Gaussians

$$\mathbf{R}\mathbf{R}^t \quad + \quad (s^2\mathbf{I} - \mathbf{R}\mathbf{R}^t) \quad = \quad s^2\,\mathbf{I}$$

To fix covariance:

- Generate perturbation vector **p** with covariance $(s^2\mathbf{I} - \mathbf{R}\mathbf{R}^t)$
- Sample spherical **z** such that $\mathbf{G}\,\mathbf{z} = \mathbf{u} - \mathbf{A}\,\mathbf{p}$
- Output $\mathbf{x} = \mathbf{p} + \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \cdot \mathbf{z}$

Check

$$\mathbf{A} \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{G}$$

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{A}\mathbf{p} + \mathbf{A}\begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \cdot \mathbf{z}$$

$$= \mathbf{A}\mathbf{p} + \mathbf{G}\mathbf{z} = \mathbf{u}$$

Takeaway for Applications

Let $\mathbf{B} \in \mathbb{Z}_q^{n \times m'}$, uniform $\mathbf{R} \in \mathbb{Z}_q^{m' \times n \log q}$, Gaussian

Let $\mathbf{A} = \left[\begin{array}{c|c} \text{B} & \text{G - BR} \end{array}\right]$

Then, $\mathbf{A}$ uniform, admits LWE and SIS inversion

$$f_{\mathbf{A}}^{-1}, \ g_{\mathbf{A}}^{-1}$$

Applications

Applications

A word about notation

Identity Based Encryption (IBE)

Identity Based Encryption (IBE)

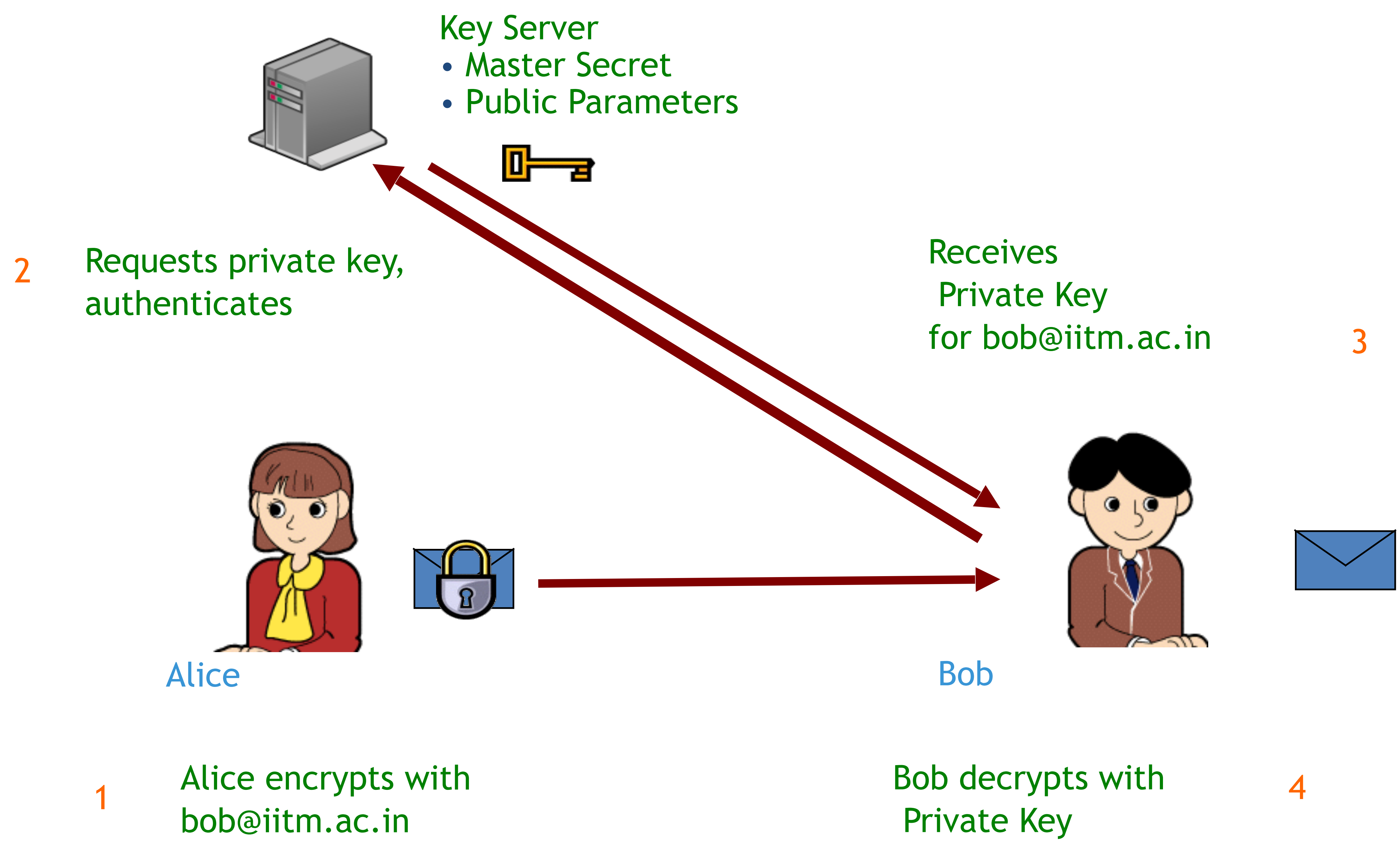
In short…………..

Identity Based Encryption (IBE)

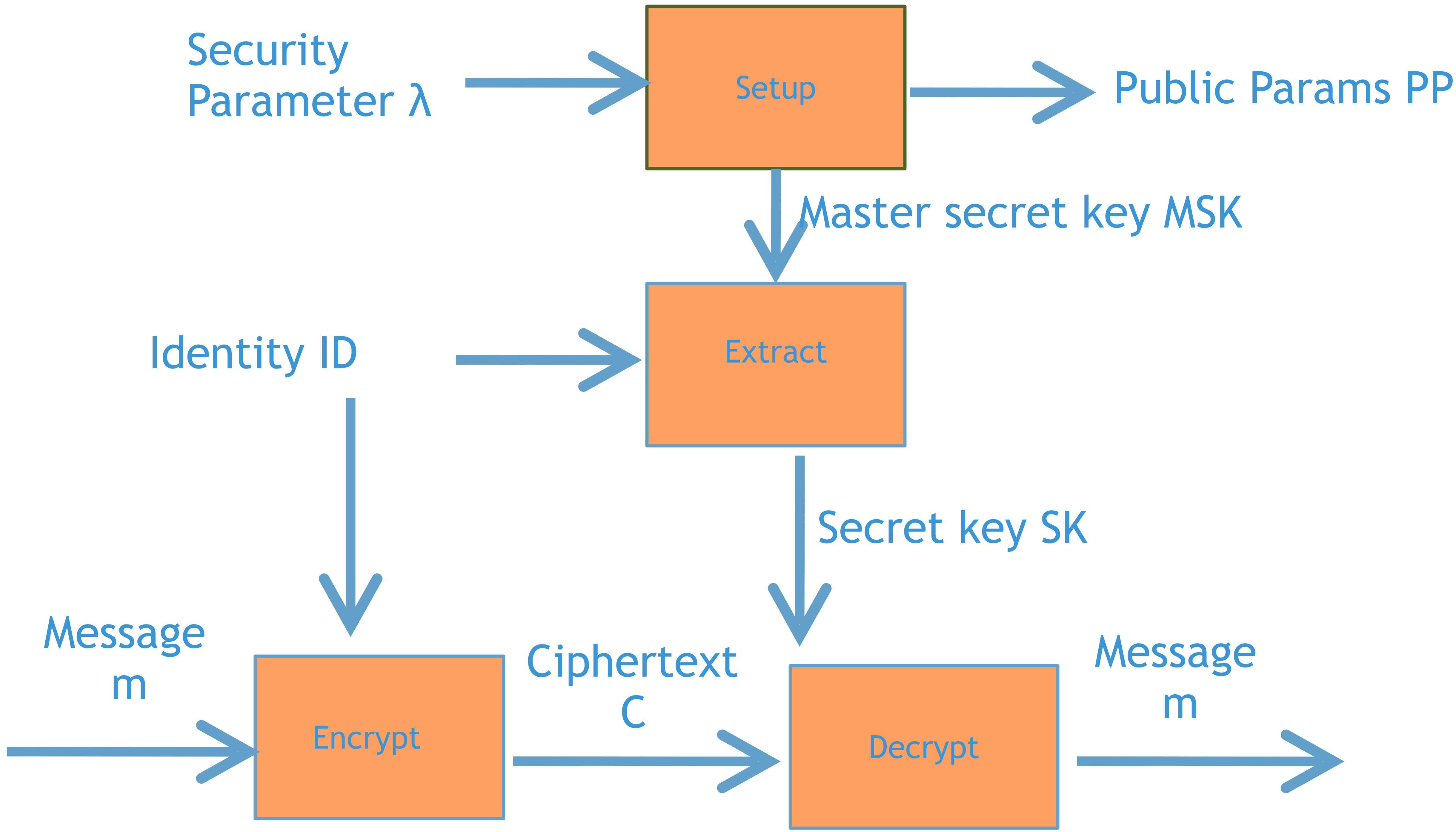
In short…………

Public Key Encryption in which ANY arbitrary string can be public key!

IBE: How does it work?



Identity Based Encryption



Bit of History

Bit of History

- Big open problem — posed in 1984 by Shamir, first solution in 2001 by Boneh and Franklin

Bit of History

- Big open problem — posed in 1984 by Shamir, first solution in 2001 by Boneh and Franklin
- First solution uses pairings
  - Beautiful solution using only CDH by Dottling & Garg (2017)

Bit of History

- Big open problem — posed in 1984 by Shamir, first solution in 2001 by Boneh and Franklin
- First solution uses pairings
  - Beautiful solution using only CDH by Dottling & Garg (2017)
- We'll see solution from lattices

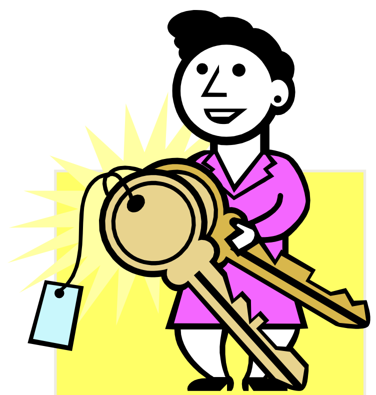
Bit of History

- Big open problem — posed in 1984 by Shamir, first solution in 2001 by Boneh and Franklin
- First solution uses pairings
    - Beautiful solution using only CDH by Dottling & Garg (2017)
- We'll see solution from lattices
- Main challenge?

Bit of History

- Big open problem — posed in 1984 by Shamir, first solution in 2001 by Boneh and Franklin
- First solution uses pairings
  - Beautiful solution using only CDH by Dottling & Garg (2017)
- We'll see solution from lattices
- Main challenge?
- Need for MSK?

IBE Security



Challenger Ch.

Adversary Ad.

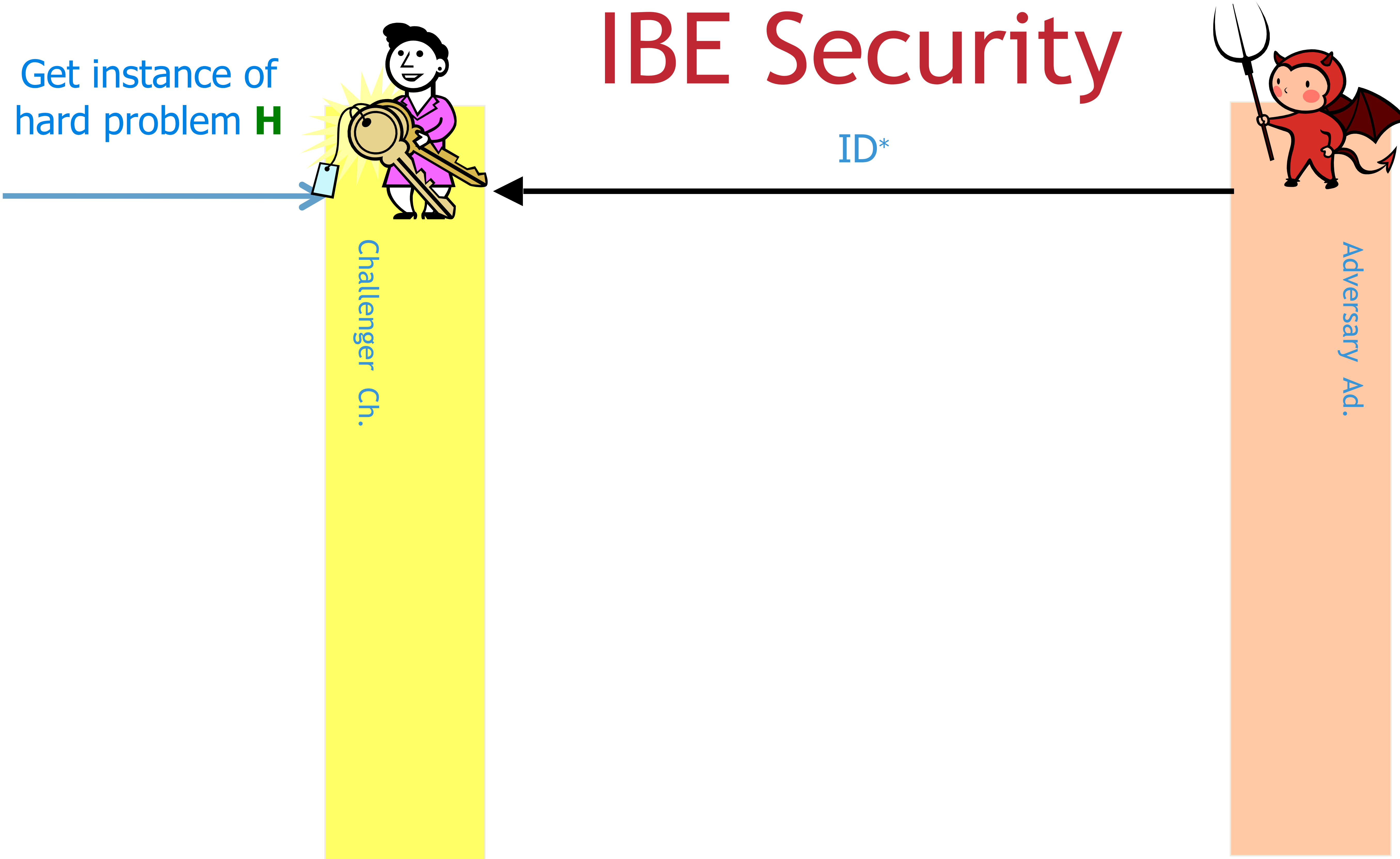
IBE Security

Get instance of
hard problem **H**

Challenger Ch.

Adversary Ad.

IBE Security



IBE Security

Get instance of hard problem **H**

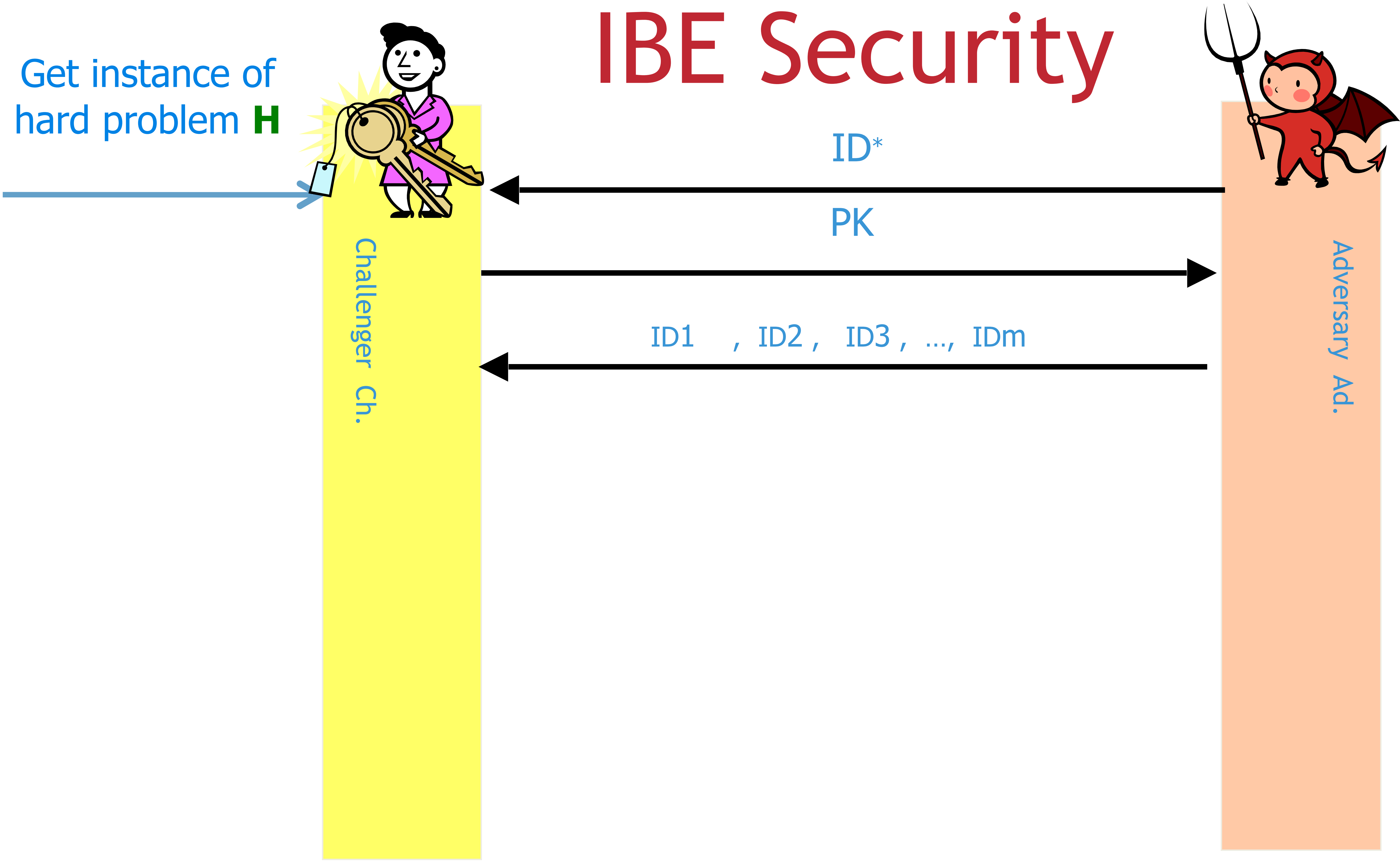
Challenger Ch.

ID*

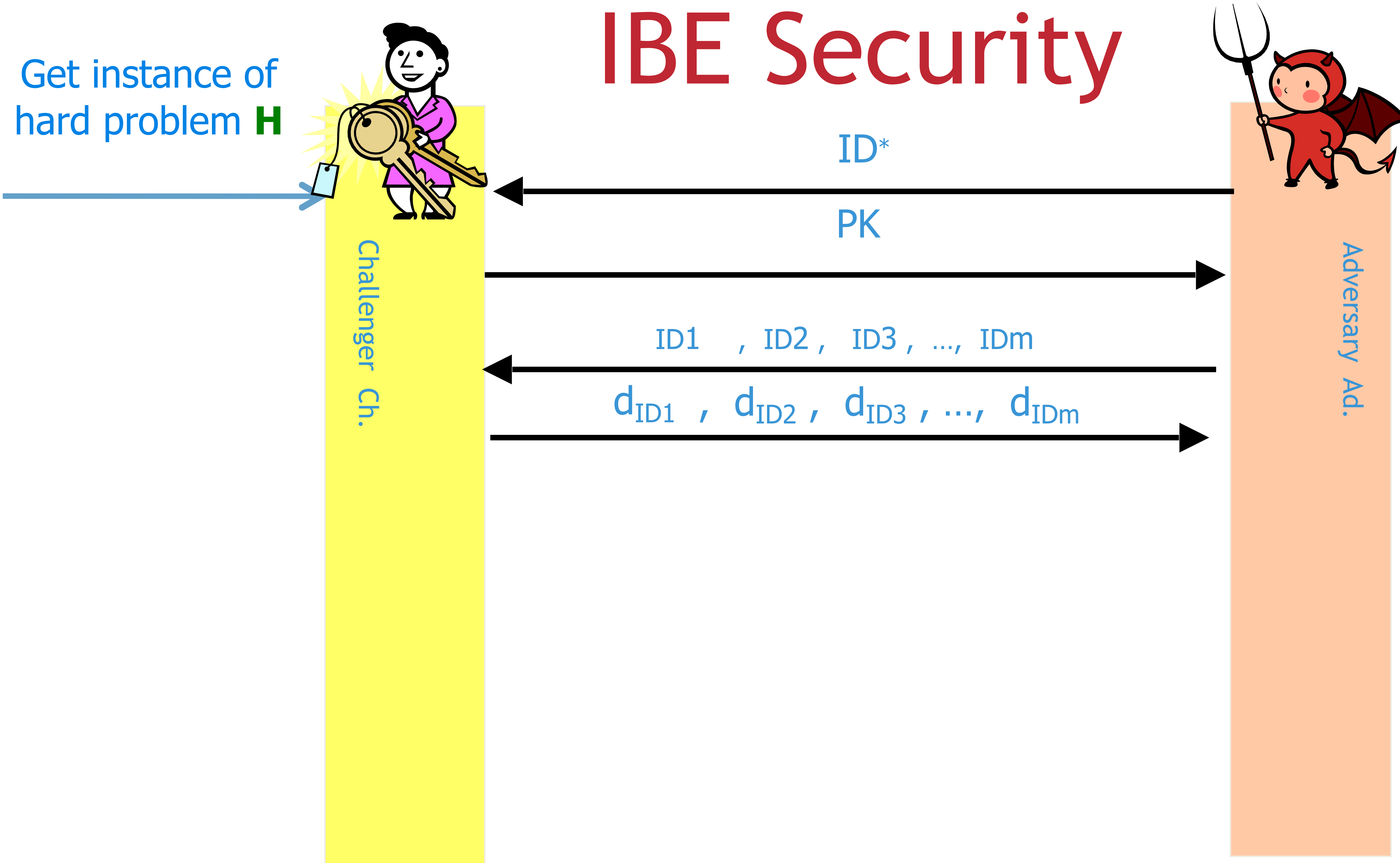
PK

Adversary Ad.

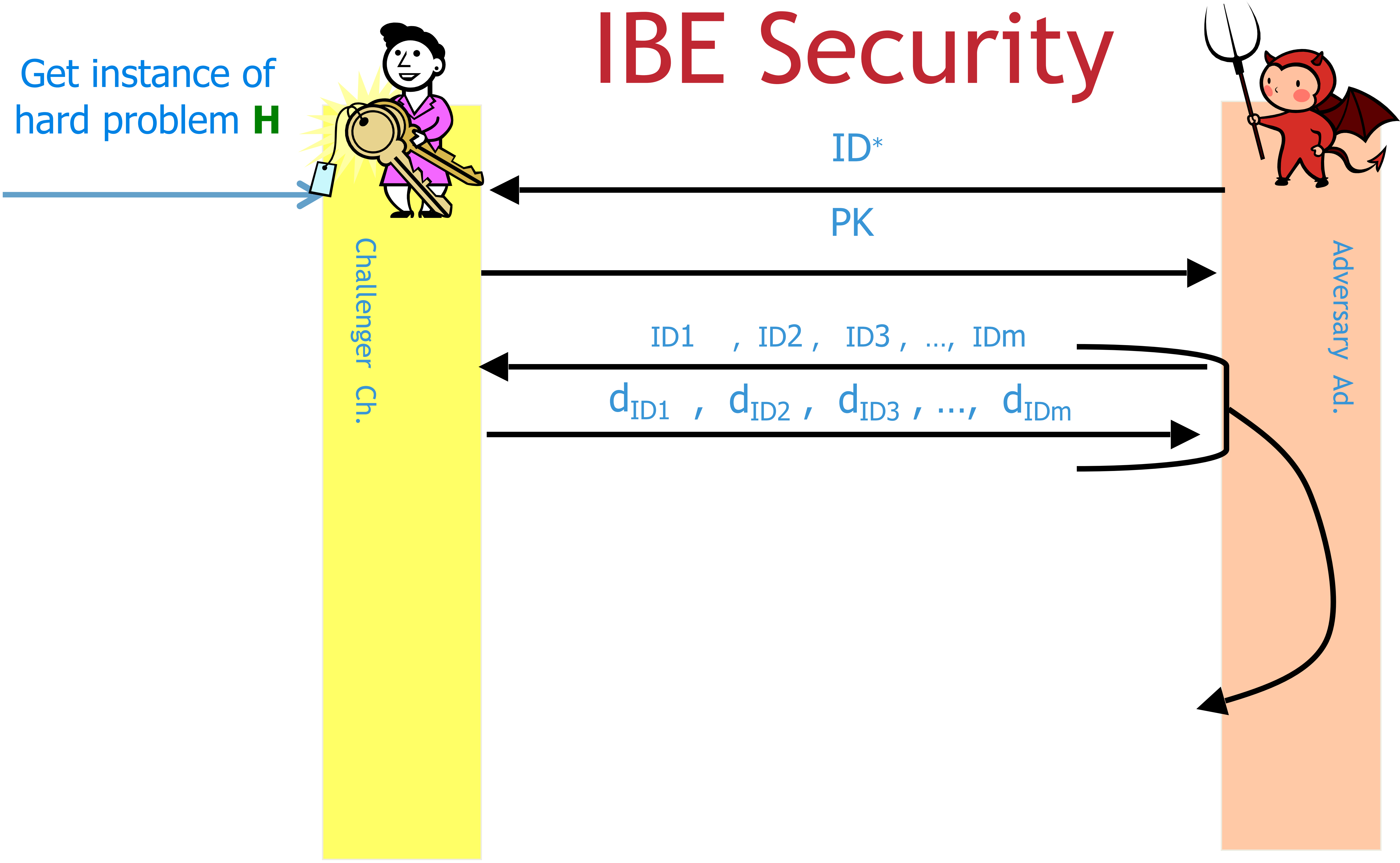
IBE Security



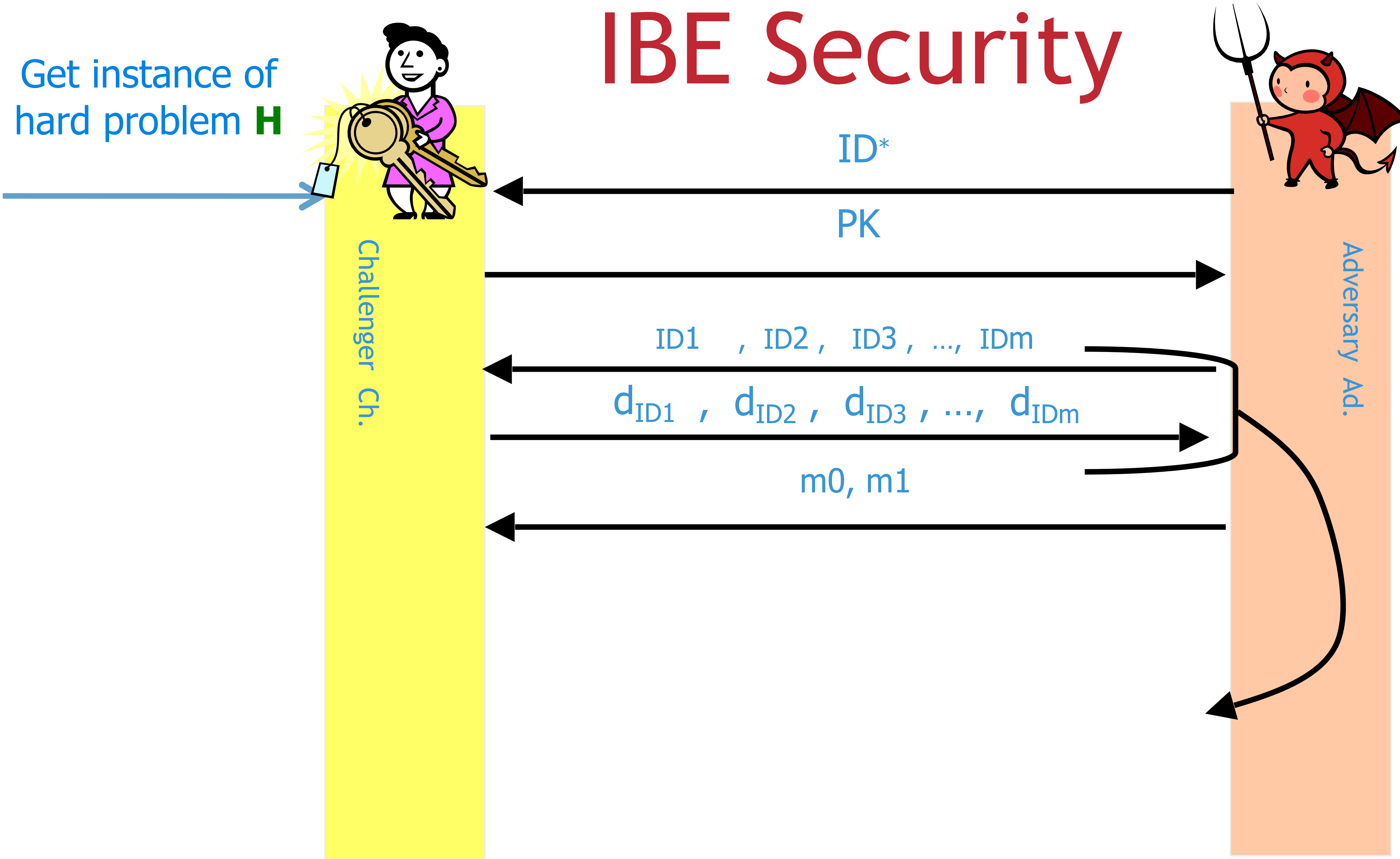
IBE Security



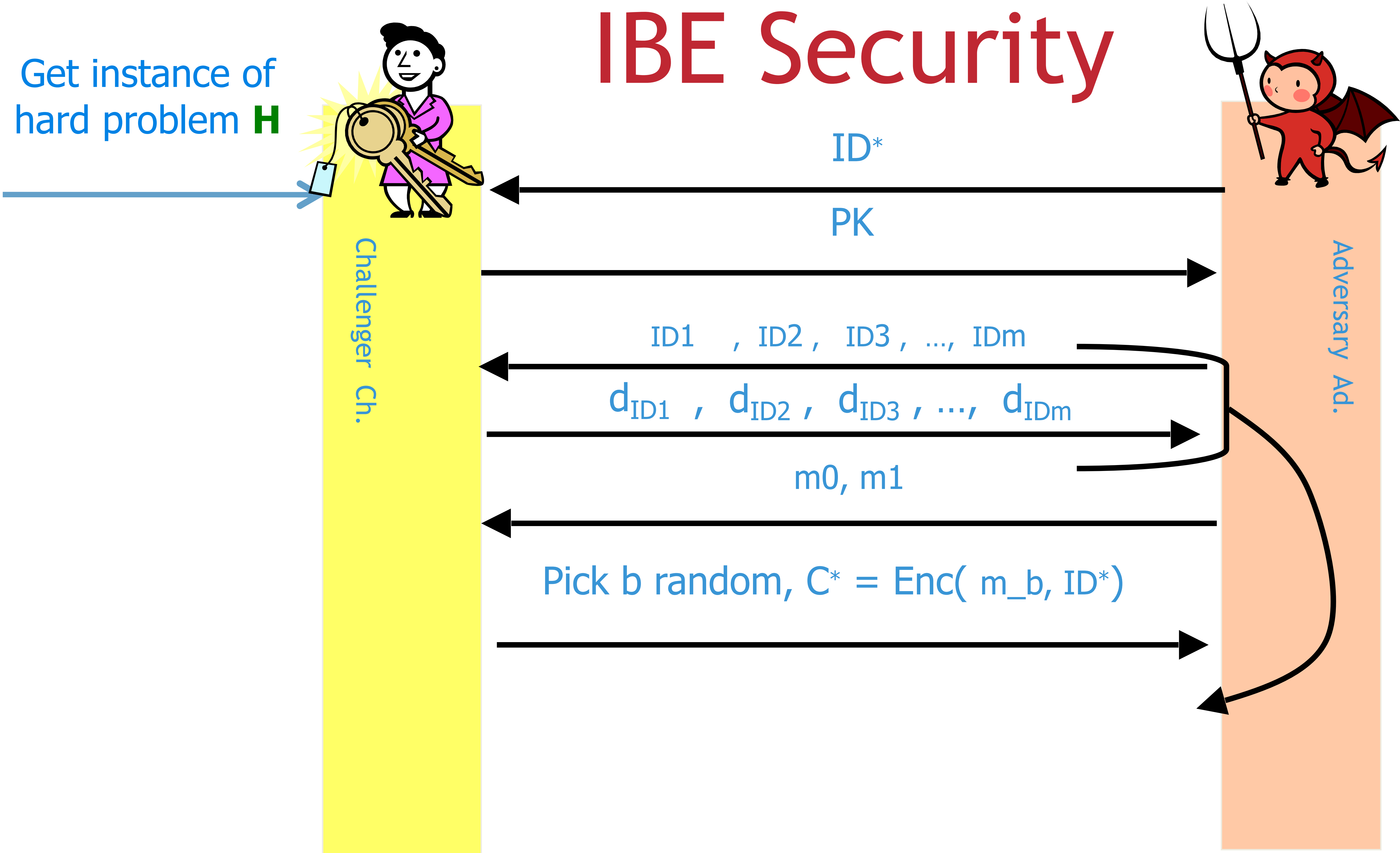
IBE Security



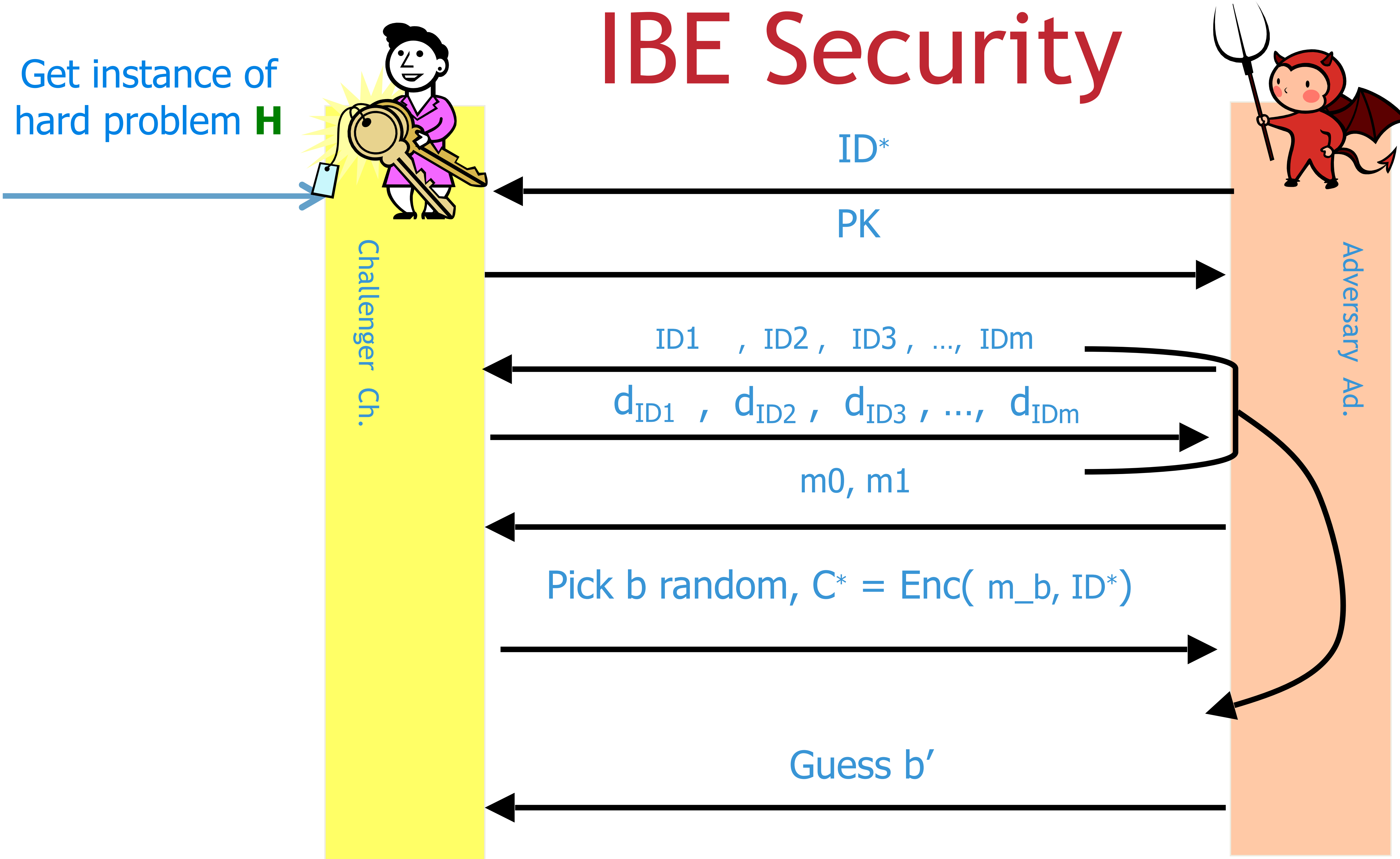
IBE Security



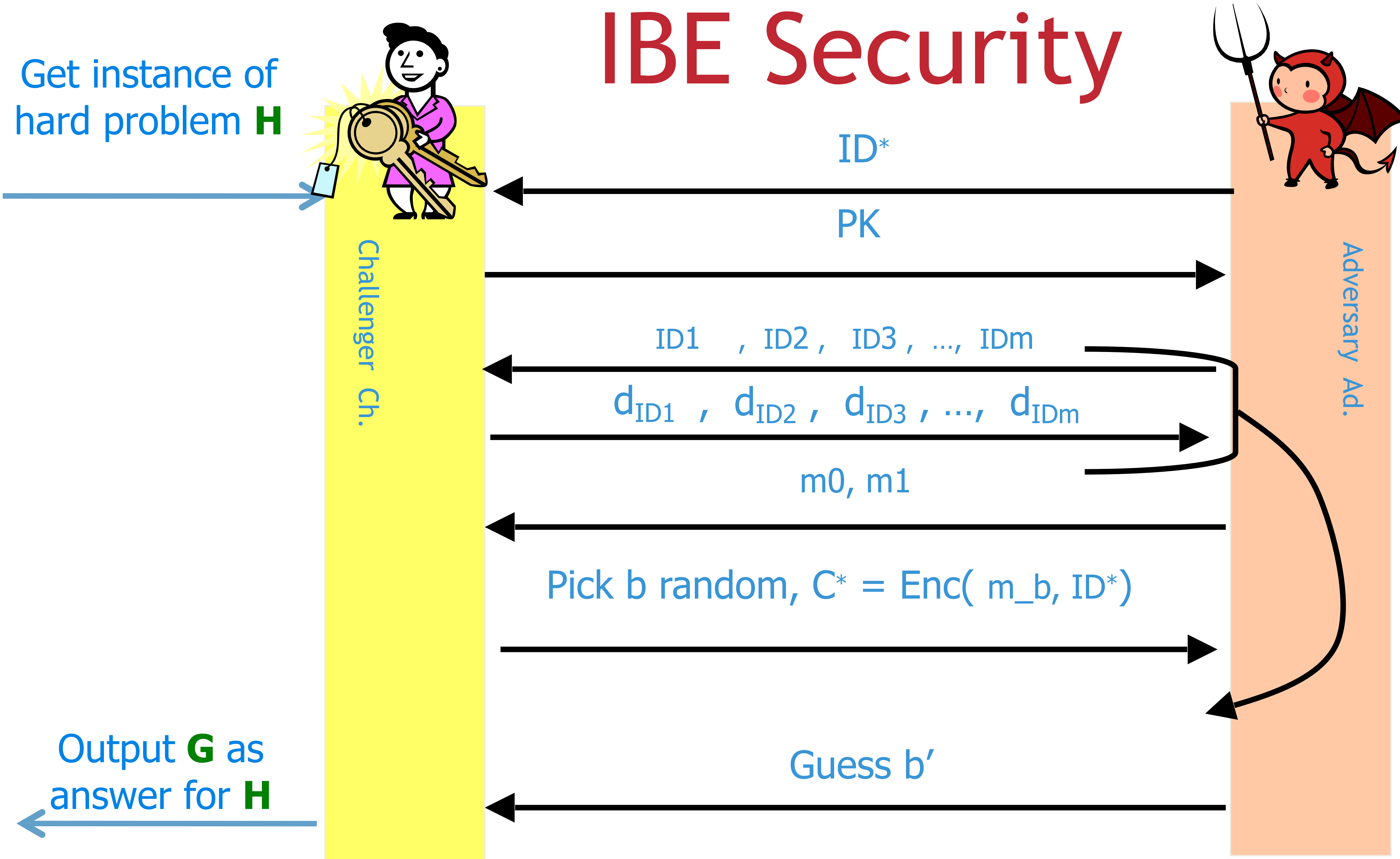
IBE Security



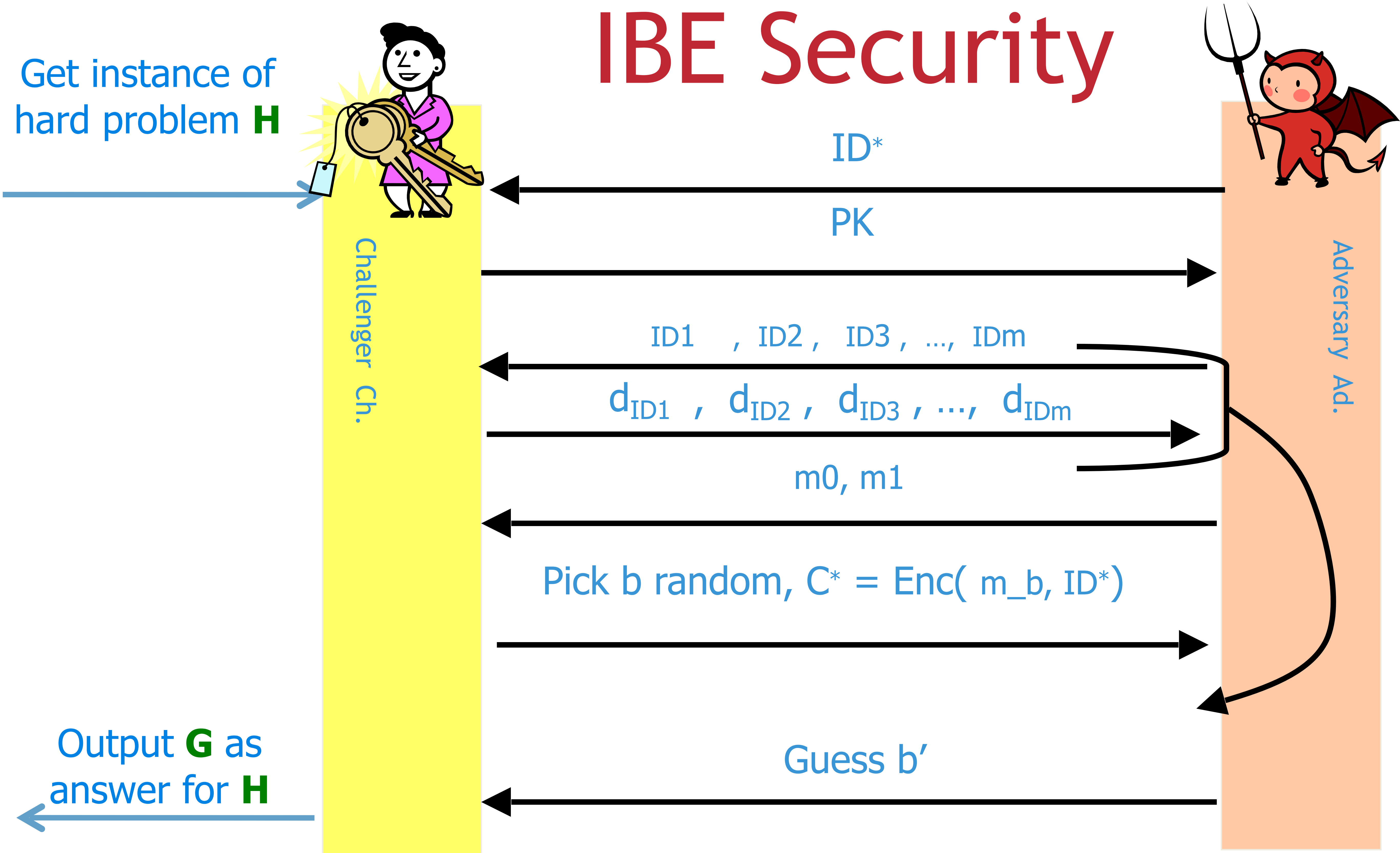
IBE Security



IBE Security



IBE Security



Attacker wins if | Pr[b=b'] - ½ | is non-negligible

Security Model: Key Points

Security Model: Key Points

Security Model: Key Points

- Ch. needs to be able to answer private key queries of Ad.

Security Model: Key Points

- Ch. needs to be able to answer private key queries of Ad.

Security Model: Key Points

- Ch. needs to be able to answer private key queries of Ad.

- Ch. should not be able to answer query for id^* (hence can't have master trapdoor)

Security Model: Key Points

- Ch. needs to be able to answer private key queries of Ad.

- Ch. should not be able to answer query for id^* (hence can't have master trapdoor)

Security Model: Key Points

- Ch. needs to be able to answer private key queries of Ad.

- Ch. should not be able to answer query for id^* (hence can't have master trapdoor)

- Ch. should be able to generate challenge ciphertext so that Ad's answer is useful.

Security Model: Key Points

- Ch. needs to be able to answer private key queries of Ad.

- Ch. should not be able to answer query for id^* (hence can't have master trapdoor)

- Ch. should be able to generate challenge ciphertext so that Ad's answer is useful.

Regev PKE

Regev PKE

- Recall $A\,e = u \bmod q$ hard to invert
- Secret: e, Public : A, u

Regev PKE

- Recall A e = u mod q hard to invert
- Secret: e, Public : A, u

$$\begin{Bmatrix} A \end{Bmatrix} \begin{bmatrix} e \end{bmatrix} \equiv \begin{bmatrix} u \end{bmatrix} \mod q$$

Regev PKE

- Recall $A\ e = u \bmod q$ hard to invert
- Secret: e, Public : A, u

$$\boxed{A}\ \boxed{e} \equiv \boxed{u} \mod q$$

- Encrypt (A, u) :
  - Pick random vector s
  - $c_0 = A^T s + \text{noise}$
  - $c_1 = u^T s + \text{noise} + \text{msg}$

Regev PKE

- Recall $A\,e = u \bmod q$ hard to invert
- Secret: e, Public : A, u

$$\boxed{A}\ \boxed{e} \equiv \boxed{u} \mod q$$

- Encrypt (A, u) :
  - Pick random vector s
  - $c_0 = A^T s + \text{noise}$
  - $c_1 = u^T s + \text{noise} + \text{msg}$
- Decrypt (e) :
  - $e^T c_0 - c_1 = \text{msg} + \text{noise}$

Regev PKE

- Recall $A\, e = u \bmod q$ hard to invert
- Secret: e, Public : A, u

$$\boxed{A}\;\boxed{e} \;\equiv\; \boxed{u} \mod q$$

- Encrypt (A, u) :
  - Pick random vector s
  - $c_0 = A^T s + \text{noise}$

Encryption matrix A

  - $c_1 = u^T s + \text{noise} + \text{msg}$
- Decrypt (e) :
  - $e^T c_0 - c_1 = \text{msg} + \text{noise}$

Regev PKE

- Recall $A\,e = u \bmod q$ hard to invert

- Secret: e, Public : A, u

$$A \; e \equiv u \quad \bmod q$$

- Encrypt (A, u) :
  - Pick random vector s
  - $c_0 = A^T s + \text{noise}$
  - $c_1 = u^T s + \text{noise} + \text{msg}$

Encryption matrix A

- Decrypt (e) :
  - $e^T c_0 - c_1 = \text{msg} + \text{noise}$

Small only if e is small

GPV IBE

GPV IBE

- Want to embed vector id in ciphertext and secret key.

GPV IBE

- Want to embed vector id in ciphertext and secret key.
- How to generate public parameters?
  - Must be independent of id (why?)
  - Must "morph" into id dependent PK for Regev

GPV IBE

- Want to embed vector id in ciphertext and secret key.
- How to generate public parameters?
  - Must be independent of id (why?)
  - Must "morph" into id dependent PK for Regev
- Let $u_{id} = H(id)$ where H is random oracle

GPV IBE

- Want to embed vector id in ciphertext and secret key.
- How to generate public parameters?
  - Must be independent of id (why?)
  - Must "morph" into id dependent PK for Regev
- Let $u_{id} = H(id)$ where H is random oracle
- Want: Perform Regev PKE with PK A, u_{id}

Random Oracle

Random Oracle

- Random oracle model assumes that well-chosen hash H (SHA3, say) behaves “like a random function”

Random Oracle

- Random oracle model assumes that well-chosen hash H (SHA3, say) behaves "like a random function"

- On any input, gives random output

Random Oracle

- Random oracle model assumes that well-chosen hash H (SHA3, say) behaves "like a random function"
- On any input, gives random output
- Repeated input, same output

Random Oracle

- Random oracle model assumes that well-chosen hash H (SHA3, say) behaves "like a random function"
- On any input, gives random output
- Repeated input, same output
- Very useful for practical schemes

Random Oracle

- Random oracle model assumes that well-chosen hash H (SHA3, say) behaves "like a random function"
- On any input, gives random output
- Repeated input, same output
- Very useful for practical schemes
- Proof in ROM allows to "program" H — gives exponential space to reduction!

GPV IBE

GPV IBE

- Recall $u_{id} = H(id)$ where H is random oracle

GPV IBE

- Recall u_{id} =H(id) where H is random oracle
- Key: small e_{id} s.t. A e_{id} = u_{id} (mod q)

GPV IBE

- Recall $u_{id} = H(id)$ where H is random oracle

- Key: small e_{id} s.t. $A\, e_{id} = u_{id}$ (mod q)

key

$$A \; e_{id} \equiv u_{id} \mod q$$

GPV IBE

- Recall u_{id} =H(id) where H is random oracle

- Key: small e_{id} s.t. A e_{id} = u_{id} (mod q)

key

$$\boxed{A}\ \boxed{e_{id}} \equiv \boxed{u_{id}} \mod q$$

How to sample?

- Construction? Proof?

GPV IBE

GPV IBE

Secret: T_A, Public : A

GPV IBE

Secret: T_A, Public : A

- Extract(T_A, id) : Set u_{id} = H(id). Find e short s.t. $A\, e_{id} = u_{id} \bmod q$

GPV IBE

Use trapdoor!

Secret: T_A, Public : A

- Extract(T_A, id) : Set u_{id} = H(id). Find e short s.t. A e_{id} = u_{id} mod q

GPV IBE

Use trapdoor!

Secret: T_A, Public : A

- Extract(T_A, id) : Set $u_{id} = H(id)$. Find e short s.t. $A e_{id} = u_{id} \bmod q$
- Encrypt (A, id) :
  - Pick random vector s
  - $c_0 = A^T s + \text{noise}$
  - $c_1 = u_{id}^T s + \text{noise} + \text{msg}$

GPV IBE

Use trapdoor!

Secret: T_A, Public : A

- Extract(T_A, id) : Set $u_{id} = H(id)$. Find e short s.t. $A e_{id} = u_{id} \bmod q$
- Encrypt (A, id) :
  - Pick random vector s
  - $c_0 = A^T s + \text{noise}$
  - $c_1 = u_{id}^T s + \text{noise} + \text{msg}$
- Decrypt (e_{id}) :
  - $e_{id}^T c_0 - c_1 = \text{msg} + \text{noise}$

Proof Idea

Proof Idea

- Selective game: reduction knows id* from beginning

Proof Idea

- Selective game: reduction knows id* from beginning
- Need:
  - Answer adversary key queries for any id $\neq$ id*
  - Unable to answer key query for id*
  - Embed LWE challenge into CT for id*

Challenge CT for id*

Challenge CT for id*

- Receive $(A, A^T s + \text{noise}), (u, u^T s + \text{noise})$ from LWE challenger

Challenge CT for id*

- Receive $(A, A^T s + \text{noise}), (u, u^T s + \text{noise})$ from LWE challenger

- "Program" H(id*) = u. Note u random so consistent with ROM!

Challenge CT for id*

- Receive (A, $A^T s$ + noise), (u, $u^T s$ + noise) from LWE challenger

- “Program" H(id*) = u. Note u random so consistent with ROM!

- Sample random bit b.

Challenge CT for id*

- Receive $(A, A^T s + \text{noise}), (u, u^T s + \text{noise})$ from LWE challenger
- "Program" H(id*) = u. Note u random so consistent with ROM!
- Sample random bit b.
- Set challenge CT as $c_0 = A^T s + \text{noise}$, $c_1 = u^T s + \text{noise} + m_b$

Challenge CT for id*

- Receive $(A, A^T s + \text{noise}), (u, u^T s + \text{noise})$ from LWE challenger
- “Program" H(id*) = u. Note u random so consistent with ROM!
- Sample random bit b.
- Set challenge CT as $c_0 = A^T s + \text{noise}$, $c_1 = u^T s + \text{noise} + m_b$
- Now, adversary sees exactly the LWE challenge: if random then b is info-theoretically hidden. No advantage!

Challenge CT for id*

- Receive $(A, A^T s + \text{noise}), (u, u^T s + \text{noise})$ from LWE challenger
- “Program" H(id*) = u. Note u random so consistent with ROM!
- Sample random bit b.
- Set challenge CT as $c_0 = A^T s + \text{noise}$, $c_1 = u^T s + \text{noise} + m_b$
- Now, adversary sees exactly the LWE challenge: if random then b is info-theoretically hidden. No advantage!
- Its success translates to success for reduction/challenger!

Key Queries

Key Queries

- Need:
  - Answer adversary key queries for any id $\neq$ id*
  - Unable to answer key query for id*

Key Queries

- Need:
  - Answer adversary key queries for any id $\neq$ id*
  - Unable to answer key query for id*

How?

Key Queries

- Need:
    - Answer adversary key queries for any id $\neq$ id*
    - Unable to answer key query for id*

How?

- Sample your own e_{id} and set $u_{id} = A\, e_{id} \bmod q$.

Key Queries

- Need:
  - Answer adversary key queries for any id $\neq$ id*
  - Unable to answer key query for id*

How?

- Sample your own e_{id} and set $u_{id} = A\, e_{id} \bmod q$.
- “Program” H(id) = u_{id}. Recall (from yesterday) u_{id} random!

Key Queries

- Need:
  - Answer adversary key queries for any $id \neq id^*$
  - Unable to answer key query for id^*

How?

- Sample your own e_{id} and set $u_{id} = A\, e_{id} \bmod q$.
- "Program" $H(id) = u_{id}$. Recall (from yesterday) u_{id} random!
- Upon hash query on id, return u_{id}.

Key Queries

- Need:
  - Answer adversary key queries for any id $\neq$ id*
  - Unable to answer key query for id*

How?

- Sample your own e_{id} and set $u_{id} = A\, e_{id} \bmod q$.
- "Program" $H(id) = u_{id}$. Recall (from yesterday) u_{id} random!
- Upon hash query on id, return u_{id}.
- Upon key query on id, return e_{id}

Standard Model?

Standard Model?

- ROM proof great first step but unrealistic

Standard Model?

- ROM proof great first step but unrealistic
- ROM cannot be instantiated [BBP03] …
  - Contrived counter-examples

Standard Model?

- ROM proof great first step but unrealistic
- ROM cannot be instantiated [BBP03] …
  - Contrived counter-examples
- Proof easy because exponential space to “program”

Standard Model?

- ROM proof great first step but unrealistic
- ROM cannot be instantiated [BBP03] …
    - Contrived counter-examples
- Proof easy because exponential space to "program"
- Can we construct it without ROM?

Standard Model

Standard Model

- Want to embed vector id in ciphertext and secret key.

Standard Model

- Want to embed vector id in ciphertext and secret key.

- Let encryption matrix F_{id} be publicly computable function of id and public parameters.

Standard Model

- Want to embed vector id in ciphertext and secret key.
- Let encryption matrix F_{id} be publicly computable function of id and public parameters.
- Perform Regev PKE with encryption matrix F_{id}

Standard Model

- Want to embed vector id in ciphertext and secret key.
- Let encryption matrix F_{id} be publicly computable function of id and public parameters.
- Perform Regev PKE with encryption matrix F_{id}
- Figure out way to compute short vector e such that

Standard Model

- Want to embed vector id in ciphertext and secret key.
- Let encryption matrix F_{id} be publicly computable function of id and public parameters.
- Perform Regev PKE with encryption matrix F_{id}
- Figure out way to compute short vector e such that

$$F_{id} \, e \equiv u \mod q$$

Std Model Identity Based Encryption [ABB10]

Std Model Identity Based Encryption [ABB10]

Parameters: $\{A_0\}$ $\{A_1\}$ $\{G\}$ u

Std Model Identity Based Encryption [ABB10]

Parameters: A_0 A_1 G u

Master Secret Key: Trapdoor for A_0

Std Model Identity Based Encryption [ABB10]

Parameters: $\left\{ A_0 \right\}$ $\left\{ A_1 \right\}$ $\left\{ G \right\}$ u

Master Secret Key: Trapdoor for A_0

KeyGen for identity id :

Std Model Identity Based Encryption [ABB10]

Parameters: $\{A_0\}$ $\{A_1\}$ $\{G\}$ u

Master Secret Key: Trapdoor for A_0

KeyGen for identity id :

Let $F_{id} = [A_0 \mid A_1 + id \times G]$

Std Model Identity Based Encryption [ABB10]

Parameters: A_0 A_1 G u

Master Secret Key: Trapdoor for A_0

KeyGen for identity id :

Let $F_{id} = [A_0 \mid A_1 + id \times G]$

key

$$F_{id} \; e \equiv u \mod q$$

Std Model Identity Based Encryption [ABB10]

Parameters: A_0 A_1 G u

Master Secret Key: Trapdoor for A_0

KeyGen for identity id :

Let $F_{id} = [A_0 \mid A_1 + id \times G]$

$$F_{id} \quad \overset{\text{key}}{e} \quad \equiv \quad u \quad \text{mod } q$$

Know how to compute trapdoor for "extended" matrix $[A_0 \mid \textit{any}\,]$

Std Model Identity Based Encryption [ABB10]

Std Model Identity Based Encryption [ABB10]

Encryption for id' = Regev PKE on matrix F_{id}

Std Model Identity Based Encryption [ABB10]

Encryption for id' = Regev PKE on matrix F_{id}

- Pick random vector s
- Let $F_{id} = [A_0 \mid A_1 + id \times G]$
- $C = u^T s + \text{noise} + \text{msg}$
- $C' = F_{id}^T s + \text{noise}$

Std Model Identity Based Encryption [ABB10]

$C_0 = u^T s + \text{noise} + m$ and $C_1 = F_{id}^T s + \text{noise}$

Std Model Identity Based Encryption [ABB10]

$C_0 = u^T s + \text{noise} + m$ and $C_1 = F_{id}^T s + \text{noise}$

Decryption : Regev decryption

Std Model Identity Based Encryption [ABB10]

$C_0 = u^T s + \text{noise} + m$ and $C_1 = F_{id}^T s + \text{noise}$

Decryption : Regev decryption

- Let $w = C_0 - e^T C_1$

Std Model Identity Based Encryption [ABB10]

$C_0 = u^T s + \text{noise} + m$ and $C_1 = F_{id}^T s + \text{noise}$

Decryption : Regev decryption

- Let $w = C_0 - e^T C_1$
- $e^T C_1 = (F_{id}\, e)^T s + \text{noise}$

Std Model Identity Based Encryption [ABB10]

$C_0 = u^T s$ + noise + m and $C_1 = F_{id}{}^T s$ + noise

Decryption : Regev decryption

- ❖ Let $w = C_0 - e^T C_1$
- ❖ $e^T C_1 = (F_{id}\, e)^T s$ + noise
- ❖ Since $F_{id}\, e = u \bmod q$, we have

Std Model Identity Based Encryption [ABB10]

$C_0 = u^T s + \text{noise} + m$ and $C_1 = F_{id}^T s + \text{noise}$

Decryption : Regev decryption

- Let $w = C_0 - e^T C_1$
- $e^T C_1 = (F_{id}\, e)^T s + \text{noise}$
- Since $F_{id}\, e = u \bmod q$, we have

$w = m + \text{noise}$ from which we can recover m.

Std Model Identity Based Encryption [ABB10]

Std Model Identity Based Encryption [ABB10]

Simulation: Let challenge identity = id^*

Std Model Identity Based Encryption [ABB10]

Simulation: Let challenge identity = id^*

- Don't have basis for A_0

Std Model Identity Based Encryption [ABB10]

Simulation: Let challenge identity = id^*

- Don't have basis for A_0
- Have basis for G

Std Model Identity Based Encryption [ABB10]

Simulation: Let challenge identity = id^*

- Don't have basis for A_0
- Have basis for G
- Let $A_1 = [A_0 R - id^* \times G]$

Std Model Identity Based Encryption [ABB10]

Simulation: Let challenge identity = id^*

- Don't have basis for A_0
- Have basis for G
- Let $A_1 = [A_0 R - id^* \times G]$

Random low norm matrix

Std Model Identity Based Encryption [ABB10]

Simulation: Let challenge identity = id*

$F_{id} = [A_0 \mid A_1 + id\, G]$

- Don't have basis for A_0
- Have basis for G
- Let $A_1 = [A_0 R - id^* \times G]$

Random low norm matrix

Std Model Identity Based Encryption [ABB10]

Simulation: Let challenge identity = id^*

$F_{id} = [A_0 \mid A_1 + id\,G]$

Random low norm matrix

- Don't have basis for A_0
- Have basis for G
- Let $A_1 = [A_0 R - id^* \times G]$
- $F_{id} = [A_0 \mid A_0 R + (id - id^*) G]$

Std Model Identity Based Encryption [ABB10]

Simulation: Let challenge identity = id*

$F_{id} = [A_0 | A_1 + id\ G]$

- Don't have basis for A_0
- Have basis for G
- Let $A_1 = [A_0 R - id^* \times G]$

Random low norm matrix

- $F_{id} = [A_0 | A_0 R + (id - id^*) G]$
- Need to find basis for F_{id} given basis for G

Std Model Identity Based Encryption [ABB10]

Simulation: Let challenge identity = id^*

$F_{id} = [A_0 | A_1 + id\, G]$

- Don't have basis for A_0
- Have basis for G
- Let $A_1 = [A_0 R - id^* \times G]$

Random low norm matrix

- $F_{id} = [A_0 | A_0 R + (id - id^*) G]$
- Need to find basis for F_{id} given basis for G

Std Model Identity Based Encryption [ABB10]

Let $\mathbf{B} \in \mathbb{Z}_q^{n\times m'}$, uniform $\mathbf{R} \in \mathbb{Z}_q^{m'\times n\log q}$, Gaussian

Let $\mathbf{A} =$

B	G - BR

Then, $\mathbf{A}$ uniform, admits LWE and SIS inversion

$$f_{\mathbf{A}}^{-1},\ g_{\mathbf{A}}^{-1}$$

Std Model Identity Based Encryption [ABB10]

MP12

Let $\mathbf{B} \in \mathbb{Z}_q^{n\times m'}$, uniform $\mathbf{R} \in \mathbb{Z}_q^{m'\times n\log q}$, Gaussian

Let $\mathbf{A} =$ | B | G - BR |

Then, $\mathbf{A}$ uniform, admits LWE and SIS inversion

$$f_{\mathbf{A}}^{-1},\ g_{\mathbf{A}}^{-1}$$

Std Model Identity Based Encryption [ABB10]

MP12

Let $\mathbf{B} \in \mathbb{Z}_q^{n \times m'}$, uniform $\mathbf{R} \in \mathbb{Z}_q^{m' \times n \log q}$, Gaussian

Let $\mathbf{A} =$

B	G - BR

Then, $\mathbf{A}$ uniform, admits LWE and SIS inversion

$$f_{\mathbf{A}}^{-1}, \; g_{\mathbf{A}}^{-1}$$

- $F_{id} = [A_0 | A_0 R + (id - id^*)G]$

Std Model Identity Based Encryption [ABB10]

MP12

Let $\mathbf{B} \in \mathbb{Z}_q^{n\times m'}$, uniform $\mathbf{R} \in \mathbb{Z}_q^{m'\times n\log q}$, Gaussian

Let $\mathbf{A} = \begin{bmatrix} \mathrm{B} & \mathrm{G - BR} \end{bmatrix}$

Then, $\mathbf{A}$ uniform, admits LWE and SIS inversion

$$f_{\mathbf{A}}^{-1},\ g_{\mathbf{A}}^{-1}$$

- $F_{id} = [A_0 | A_0 R + (id - id^*) G]$
- Can find basis for F_{id} given basis for G !

Std Model Identity Based Encryption [ABB10]

MP12

Let $\mathbf{B} \in \mathbb{Z}_q^{n\times m'}$, uniform $\mathbf{R} \in \mathbb{Z}_q^{m'\times n\log q}$, Gaussian

Let $\mathbf{A} = \begin{bmatrix} B & G - BR \end{bmatrix}$

Then, $\mathbf{A}$ uniform, admits LWE and SIS inversion

$$f_{\mathbf{A}}^{-1},\ g_{\mathbf{A}}^{-1}$$

- $F_{id} = [A_0 | A_0R + (id - id^*)G]$
- Can find basis for F_{id} given basis for G !

Developed in ABB10

Std Model Identity Based Encryption [ABB10]

MP12

Let $\mathbf{B} \in \mathbb{Z}_q^{n \times m'}$, uniform $\mathbf{R} \in \mathbb{Z}_q^{m' \times n \log q}$, Gaussian

Let $\mathbf{A} =$ | B | G - BR |

Then, $\mathbf{A}$ uniform, admits LWE and SIS inversion

$$f_{\mathbf{A}}^{-1}, \; g_{\mathbf{A}}^{-1}$$

Developed in ABB10

- $F_{id} = [A_0 | \; A_0 R + (id - id^*) G]$
- Can find basis for F_{id} given basis for G !
- Trapdoor vanishes for id = id*

Std Model Identity Based Encryption [ABB10]

Real System

Simulation

Std Model Identity Based Encryption [ABB10]

$$PP = A_0, A_1, G$$

Real System

Simulation

Std Model Identity Based Encryption [ABB10]

$$PP = A_0, A_1, G$$

Real System | Simulation

MSK = Trapdoor for A_0

Std Model Identity Based Encryption [ABB10]

$$PP = A_0, A_1, G$$

Real System

MSK = Trapdoor for A_0

Simulation

MSK = R

Std Model Identity Based Encryption [ABB10]

$$\mathrm{PP} = A_0, A_1, G$$

Real System

MSK = Trapdoor for A_0

A_1 = Randomly chosen

Simulation

MSK = R

Std Model Identity Based Encryption [ABB10]

$$PP = A_0, A_1, G$$

Real System

MSK = Trapdoor for A_0

A_1 = Randomly chosen

Simulation

MSK = R

A_1 = $A_0R - id^* G$

Std Model Identity Based Encryption [ABB10]

$$PP = A_0, A_1, G$$

Real System		Simulation	
MSK	= Trapdoor for A_0	MSK	= R
A_1	= Randomly chosen	A_1	$= A_0R - id^* G$

Indistinguishable since R is random!

Std Model Identity Based Encryption [ABB10]

$$PP = A_0, A_1, G$$

Real System

MSK = Trapdoor for A_0

A_1 = Randomly chosen

Simulation

MSK = R

A_1 = $A_0R - id^* G$

Indistinguishable since R is random!

Encryption matrix $F_{id} = [A_0|A_1+id.G]$

Std Model Identity Based Encryption [ABB10]

$$PP = A_0, A_1, G$$

Real System		Simulation	
MSK	= Trapdoor for A_0	MSK	= R
A_1	= Randomly chosen	A_1	$= A_0R - id^* G$

Indistinguishable since R is random!

Real System

Encryption matrix $F_{id} = [A_0 | A_1 + id.G]$

Simulation

Encryption matrix $F_{id} = [A_0 \mid A_1 + id.G]$

$= [A_0 \mid A_0R + (id - id^*)G]$

Std Model Identity Based Encryption [ABB10]

$$PP = A_0, A_1, G$$

Real System

MSK = Trapdoor for A_0

A_1 = Randomly chosen

Encryption matrix $F_{id} = [A_0 | A_1 + id.G]$

Secret Key = short vector in F_{id}

Simulation

MSK = R

$A_1 = A_0R - id^* G$

Encryption matrix $F_{id} = [A_0 \mid A_1 + id.G]$

$= [A_0 \mid A_0R + (id - id^*)G]$

Indistinguishable since R is random!

Std Model Identity Based Encryption [ABB10]

$$PP = A_0, A_1, G$$

Real System

MSK = Trapdoor for A_0

A_1 = Randomly chosen

Encryption matrix $F_{id} = [A_0 | A_1 + id.G]$

Secret Key = short vector in F_{id}

Simulation

MSK = R

$A_1 = A_0R - id^* G$

Encryption matrix $F_{id} = [A_0 \mid A_1 + id.G]$

$= [A_0 \mid A_0R + (id - id^*)G]$

Secret Key = short vector in F_{id}

Indistinguishable since R is random!

Std Model Identity Based Encryption [ABB10]

$$PP = A_0, A_1, G$$

Real System

MSK = Trapdoor for A_0

A_1 = Randomly chosen

Encryption matrix $F_{id} = [A_0 | A_1 + id.G]$

Secret Key = short vector in F_{id}

MSK ➔ Key for any id

Simulation

MSK = R

$A_1 = A_0R - id^* G$

Encryption matrix $F_{id} = [A_0 \mid A_1 + id.G]$

$= [A_0 \mid A_0R + (id - id^*)G]$

Secret Key = short vector in F_{id}

Indistinguishable since R is random!

Std Model Identity Based Encryption [ABB10]

$$PP = A_0, A_1, G$$

Real System

MSK = Trapdoor for A_0

A_1 = Randomly chosen

Encryption matrix $F_{id} = [A_0 | A_1 + id.G]$

Secret Key = short vector in F_{id}

MSK ➔ Key for any id

Simulation

MSK = R

$A_1 = A_0R - id^* G$

Encryption matrix $F_{id} = [A_0 \mid A_1 + id.G]$

$= [A_0 \mid A_0R + (id - id^*)G]$

Secret Key = short vector in F_{id}

Trapdoor for G ➔ Key for id ≠ id*

Indistinguishable since R is random!

The matrix R

- Matrix R : each column randomly and independently chosen from $\{+1, -1\}^m$
- (A_0, A_1) indistinguishable from (A_0, A_0R)

  by leftover hash lemma

- Roughly states that R has enough entropy to make A_0R look like A_1

Generalizing to inner products (AFV11)

Generalizing to Inner Product (KSW08)

Key : $y = (y_1, \ldots, y_n)$

CT : $x = (x_1, \ldots, x_n)$

$$\text{Function } f(x, y) = \begin{cases} 1 & \text{If } \langle x \,.\, y\rangle = 0 \\ 0 & \text{otherwise} \end{cases}$$

Generalizing to Inner Product (KSW08)

Key : $y = (y_1, \ldots, y_n)$

CT : $x = (x_1, \ldots, x_n)$

Function $f(x, y)$ = 1 If $\langle x \cdot y \rangle = 0$

0 otherwise

Supports:

- OR –- Bob OR Alice $OR_{A,B}(z) = 1$ if $z = A \; OR \; z = B$

$$p(z) = (A - z)(B - z)$$

- CNF/DNF formulas of bounded size

Generalizing to Inner Product (KSW08)

Key : $y = (y_1, \ldots, y_n)$

CT : $x = (x_1, \ldots, x_n)$

Ciphertext Hides Attributes x_i

Function $f(x, y) = 1$ If $\langle x \,.\, y\rangle = 0$

0 otherwise

Supports:

- OR –- Bob OR Alice $OR_{A,B}(z) = 1$ **if** $z = A \;OR\; z = B$

$$p(z) = (A - z)(B - z)$$

- CNF/DNF formulas of bounded size

Generalizing to Inner Product (AFV11)

Generalizing to Inner Product (AFV11)

- Parameters for |x| = |y| = 4:

$\{A_1\}\ \{A_2\}\ \{A_3\}\ \{A_4\} \qquad \{A\} \qquad u$

Generalizing to Inner Product (AFV11)

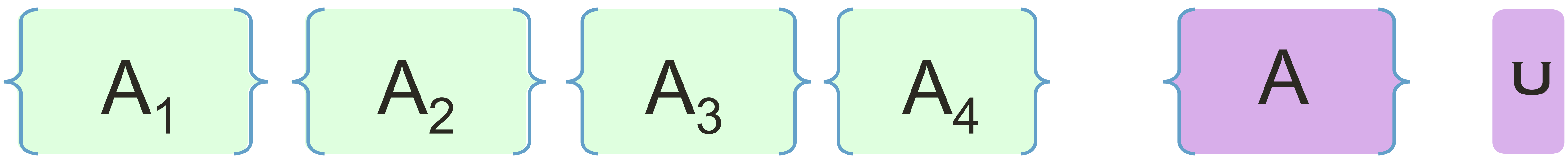
- Parameters for $|x| = |y| = 4$:

$$A_1 \quad A_2 \quad A_3 \quad A_4 \qquad A \qquad u$$

- Master Secret Key: Trapdoor for A

Generalizing to Inner Product (AFV11)

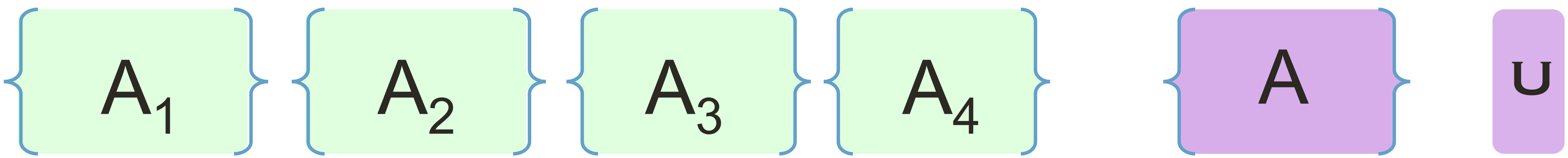
- Parameters for |x| = |y| = 4:



- Master Secret Key: Trapdoor for A
- Define $F_y = [A \mid \Sigma y_i A_i\,]$

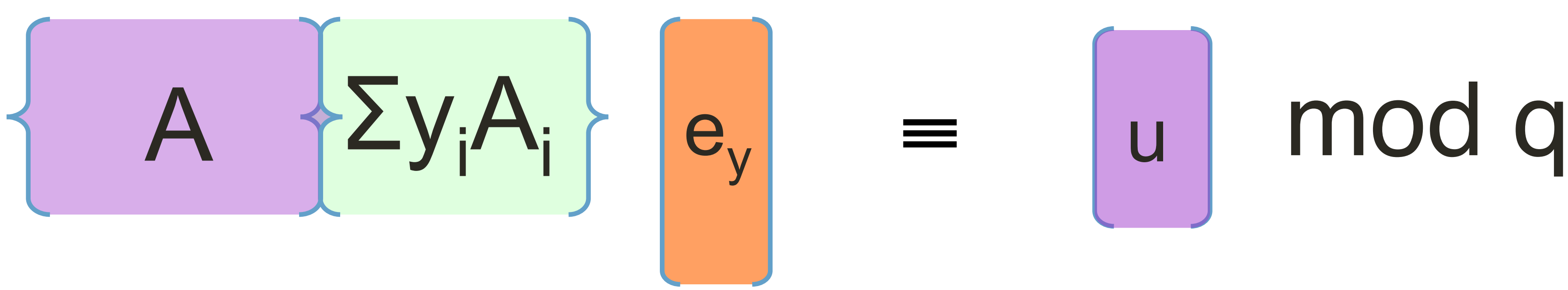
Generalizing to Inner Product (AFV11)

- Parameters for |x| = |y| = 4:



- Master Secret Key: Trapdoor for A

- Define $F_y = [A \mid \Sigma y_i A_i]$



Generalizing to Inner Product (AFV11)

- Parameters for |x| = |y| = 4:

$$A_1 \quad A_2 \quad A_3 \quad A_4 \qquad A \qquad u$$

- Master Secret Key: Trapdoor for A

- Define $F_y = [A \mid \Sigma y_i A_i]$

$$\left[\, A \mid \Sigma y_i A_i \,\right] \, e_y \equiv u \mod q$$

key

Generalizing to Inner Product (AFV11)

Generalizing to Inner Product (AFV11)

Encryption for vector $x = (x_1\ x_2\ x_3\ x_4)$:

Generalizing to Inner Product (AFV11)

Encryption for vector $x = (x_1\ x_2\ x_3\ x_4)$:

- ❖ Pick random vector s
- ❖ $C = u^T s + \text{noise} + \text{msg}$
- ❖ $C' = A^T s + \text{noise}$

Generalizing to Inner Product (AFV11)

Encryption for vector $x = (x_1\ x_2\ x_3\ x_4)$:

- ❖ Pick random vector s
- ❖ $C\ = u^T s + \text{noise} + \text{msg}$
- ❖ $C' = A^T s + \text{noise}$
- ❖ Set $C_i = (A_i + x_i\ G)^T\ s + \text{noise}$

Generalizing to Inner Product (AFV11)

Decryption

(CT_x, SK_y) :

Generalizing to Inner Product (AFV11)

Decryption (CT_x, SK_y) :

$$C_i = (A_i + x_i\, G)^T s + \text{noise}$$

Generalizing to Inner Product (AFV11)

Decryption (CT_x, SK_y) :

$C_i = (A_i + x_i\ G)^T\ s + \text{noise}$

$C' = A^T\ s + \text{noise}$

Generalizing to Inner Product (AFV11)

Decryption (CT_x, SK_y) :

$C_i = (A_i + x_i\, G)^T s + \text{noise}$

$C' = A^T s + \text{noise}$

$$\left[\, A \;\middle|\; \Sigma y_i A_i \,\right] \, e_y \equiv u \mod q$$

Generalizing to Inner Product (AFV11)

Decryption (CT_x, SK_y) :

$C_i = (A_i + x_i\, G)^T s + \text{noise}$

$C' = A^T s + \text{noise}$

$$\begin{bmatrix} A & \Sigma y_i A_i \end{bmatrix} \begin{bmatrix} e_y \end{bmatrix} \equiv \begin{bmatrix} u \end{bmatrix} \mod q$$

Set $C_y = \Sigma\, y_i\, C_i$

$$= (\Sigma\, y_i\, A_i + \Sigma\, y_i\, x_i\, G)^T s + \Sigma\, y_i \text{ noise}$$

Generalizing to Inner Product (AFV11)

Decryption (CT_x, SK_y) :

$$C_i = (A_i + x_i\, G)^T s + \text{noise}$$

$$C' = A^T s + \text{noise}$$

$$\begin{bmatrix} A & \Sigma y_i A_i \end{bmatrix} \begin{bmatrix} e_y \end{bmatrix} \equiv \begin{bmatrix} u \end{bmatrix} \text{ mod } q$$

Set $C_y = \Sigma\, y_i\, C_i$

$$= (\Sigma\, y_i\, A_i + \Sigma\, \cancel{y_i\, x_i}\, G\,)^T s + \Sigma\, y_i \text{ noise}$$

Generalizing to Inner Product (AFV11)

Decryption (CT_x, SK_y) :

$$C_i = (A_i + x_i\, G)^T s + \text{noise}$$

$$C' = A^T s + \text{noise}$$

$$\left[\, A \;\middle|\; \Sigma y_i A_i \,\right] e_y \equiv u \mod q$$

Set $C_y = \Sigma\, y_i\, C_i$

$$= (\Sigma\, y_i\, A_i + \Sigma\, \cancel{y_i\, x_i}\, G)^T s + \Sigma\, y_i\, \text{noise}$$

$$[\, C' | C_y \,] = [A \mid \Sigma\, y_i\, A_i\,]^T s + \text{noise}$$

Generalizing to Inner Product (AFV11)

Decryption (CT_x, SK_y) :

$$C_i = (A_i + x_i\, G)^T s + \text{noise}$$

$$C' = A^T s + \text{noise}$$

$$\left[\, A \mid \Sigma y_i A_i \,\right] \, e_y \equiv u \mod q$$

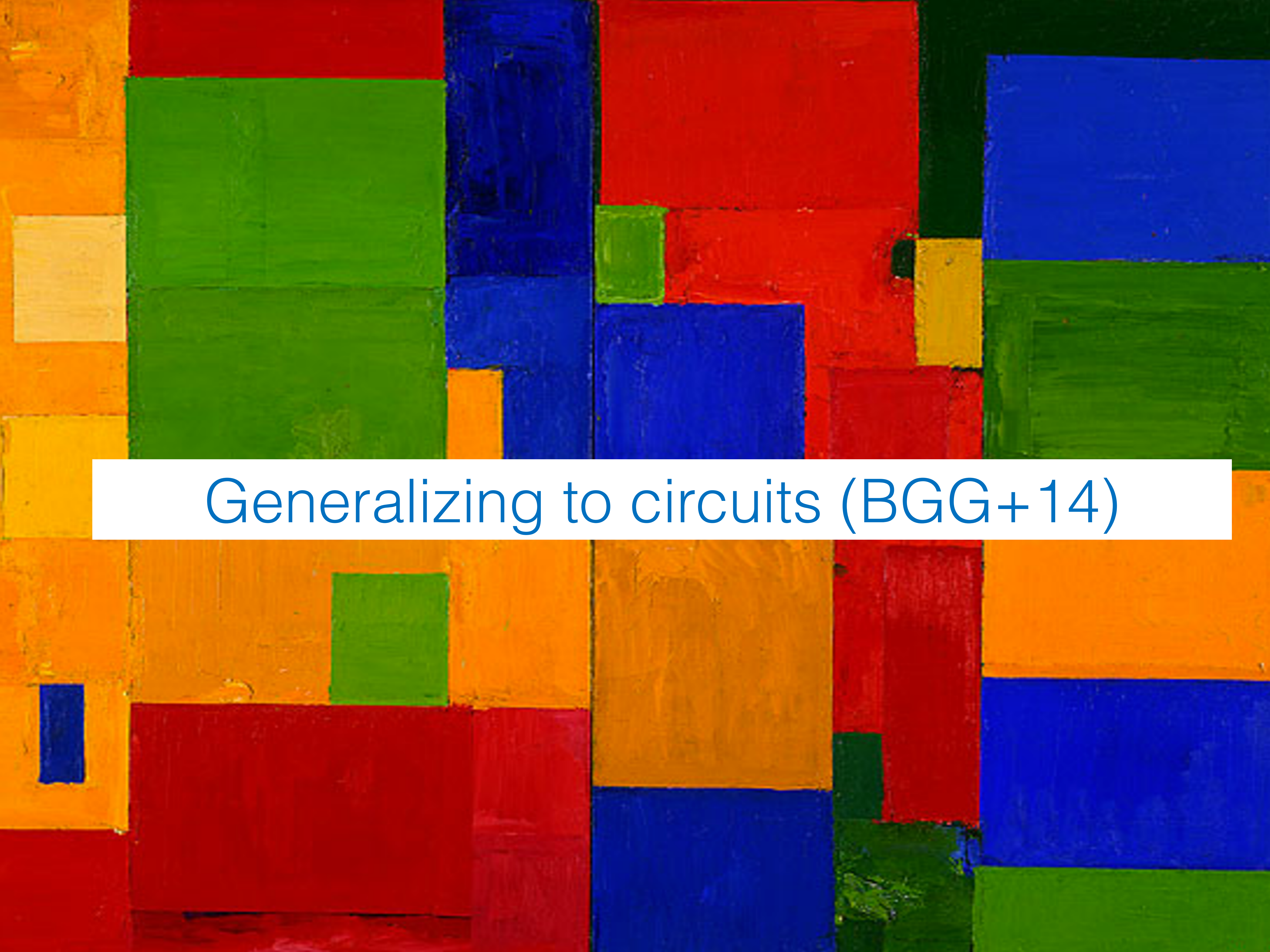
Set $C_y = \Sigma\, y_i\, C_i$

$$= (\Sigma\, y_i\, A_i + \Sigma\, \cancel{y_i x_i}\, G\,)^T s + \Sigma\, y_i\, \text{noise}$$

$$[\, C' | C_y \,] = [A \mid \Sigma\, y_i\, A_i\,]^T s + \text{noise}$$

But this is what we have the key for !

Perform Regev Decryption.



Generalizing to circuits (BGG+14)

Recall Ciphertext Structure

Recall Ciphertext Structure

Encryption for vector $x = (x_1\ x_2\ x_3\ x_4)$:

Recall Ciphertext Structure

Encryption for vector $x = (x_1\ x_2\ x_3\ x_4)$:

$C = u^T s + \text{noise} + \text{msg}$, $C' = A^T s + \text{noise}$

Recall Ciphertext Structure

Encryption for vector $x = (x_1\ x_2\ x_3\ x_4)$:

$C = u^T s + \text{noise} + \text{msg}, \quad C' = A^T s + \text{noise}$

$C_i = (A_i + x_i G)^T s + \text{noise}$

Recall Ciphertext Structure

Encryption for vector $x = (x_1\ x_2\ x_3\ x_4)$:

$C = u^T s + \text{noise} + \text{msg}, \quad C' = A^T s + \text{noise}$

$C_i = (A_i + x_i G)^T s + \text{noise}$

Previously: Could evaluate on CT to obtain

Recall Ciphertext Structure

Encryption for vector $x = (x_1\ x_2\ x_3\ x_4)$:

$C = u^T s + \text{noise} + \text{msg}, \quad C' = A^T s + \text{noise}$

$C_i = (A_i + x_i\, G)^T s + \text{noise}$

Previously: Could evaluate on CT to obtain

$C_{<x,\, y>} = (A_y + \langle x, y \rangle\, G)^T s + \text{noise}$

Recall Ciphertext Structure

Encryption for vector $x = (x_1\ x_2\ x_3\ x_4)$:

$C\ = u^T s + \text{noise} + \text{msg}, \quad C' = A^T s + \text{noise}$

$C_i = (A_i + x_i\ G)^T\ s + \text{noise}$

Previously: Could evaluate on CT to obtain

$C_{<x,\ y>} = (A_y + <x,\ y>\ G)^T\ s + \text{noise}$

When <x, y> = 0, obtain CT that encodes f alone,
Keygen may compute matching key

Recall Ciphertext Structure

Encryption for vector $x = (x_1\ x_2\ x_3\ x_4)$:

$C\ = u^T s + \text{noise} + \text{msg},\ \ C' = A^T s + \text{noise}$

$C_i = (A_i + x_i\, G)^T s + \text{noise}$

Previously: Could evaluate on CT to obtain

$C_{<x,\, y>} = (A_y + \langle x, y\rangle\, G)^T s + \text{noise}$

When <x, y> = 0, obtain CT that encodes f alone,
Keygen may compute matching key

Generalize to arbitrary f?

Recall Ciphertext Structure

Encryption for vector $x = (x_1\ x_2\ x_3\ x_4)$:

$C = u^T s + \text{noise} + \text{msg}, \quad C' = A^T s + \text{noise}$

$C_i = (A_i + x_i\, G)^T s + \text{noise}$

Previously: Could evaluate on CT to obtain

$C_{\langle x, y\rangle} = (A_y + \langle x, y\rangle\, G)^T s + \text{noise}$

When <x, y> = 0, obtain CT that encodes f alone,
Keygen may compute matching key

Generalize to arbitrary f?

$C_{f(x)} = (A_f + f(x)\, G)^T s + \text{noise}$

Handling Multiplication [BGG+14]

Handling Multiplication [BGG+14]

$C_1 = (A_1 + x_1\, G)^T\, s + \text{noise}$

Handling Multiplication [BGG+14]

$C_1 = (A_1 + x_1 G)^T s + \text{noise}$ $\quad$ $C_2 = (A_2 + x_2 G)^T s + \text{noise}$

Handling Multiplication [BGG+14]

$C_1 = (A_1 + x_1\, G)^T s + \text{noise}$ $\quad$ $C_2 = (A_2 + x_2\, G)^T s + \text{noise}$

Want $\quad C_{x1\ x2} = (A_{12} + x_1 x_2\, G)^T s + \text{noise}$

Handling Multiplication [BGG+14]

$C_1 = (A_1 + x_1 G)^T s + \text{noise}$ $\quad$ $C_2 = (A_2 + x_2 G)^T s + \text{noise}$

Want $\quad C_{x1\ x2} = (A_{12} + x_1 x_2 G)^T s + \text{noise}$

Key Observation: x may be used in evaluation !

Handling Multiplication [BGG+14]

$C_1 = (A_1 + x_1 G)^T s + \text{noise}$ $\quad$ $C_2 = (A_2 + x_2 G)^T s + \text{noise}$

Want $\quad C_{x1\ x2} = (A_{12} + x_1 x_2 G)^T s + \text{noise}$

Key Observation: x may be used in evaluation !

$(A_1 + x_1 G)$

Handling Multiplication [BGG+14]

$C_1 = (A_1 + x_1 G)^T s + \text{noise}$ $C_2 = (A_2 + x_2 G)^T s + \text{noise}$

Want $C_{x1\ x2} = (A_{12} + x_1 x_2 G)^T s + \text{noise}$

Key Observation: x may be used in evaluation !

$(A_1 + x_1 G)$

$(A_2 + x_2 G)$

Handling Multiplication [BGG+14]

$C_1 = (A_1 + x_1\, G)^T s + \text{noise}$ $\qquad$ $C_2 = (A_2 + x_2\, G)^T s + \text{noise}$

Want $\quad C_{x1\ x2} = (A_{12} + x_1 x_2\, G)^T s + \text{noise}$

Key Observation: x may be used in evaluation !

$(A_1 + x_1\, G)\ G^{-1}\,(-A_2)$

$(A_2 + x_2\, G)$

Handling Multiplication [BGG+14]

Recall $G\, G^{-1}(A) = A$

$C_1 = (A_1 + x_1 G)^T s + \text{noise}$ $\quad$ $C_2 = (A_2 + x_2 G)^T s + \text{noise}$

Want $\quad C_{x1\ x2} = (A_{12} + x_1 x_2 G)^T s + \text{noise}$

Key Observation: x may be used in evaluation !

$(A_1 + x_1 G)\ G^{-1}(-A_2)$

$(A_2 + x_2 G)$

Handling Multiplication [BGG+14]

Recall $G\, G^{-1}(A) = A$

$C_1 = (A_1 + x_1 G)^T s + \text{noise}$ $\quad C_2 = (A_2 + x_2 G)^T s + \text{noise}$

Want $\quad C_{x1\ x2} = (A_{12} + x_1 x_2 G)^T s + \text{noise}$

Key Observation: x may be used in evaluation !

$(A_1 + x_1 G)\ G^{-1}(-A_2)$

$(A_2 + x_2 G)\ (x_1)$

Handling Multiplication [BGG+14]

Recall $G\, G^{-1}(A) = A$

$C_1 = (A_1 + x_1 G)^T s + \text{noise}$ $\quad$ $C_2 = (A_2 + x_2 G)^T s + \text{noise}$

Want $\quad C_{x1\ x2} = (A_{12} + x_1 x_2 G)^T s + \text{noise}$

Key Observation: x may be used in evaluation !

$(A_1 + x_1 G)\ G^{-1}(-A_2) \quad = (A_1 G^{-1}(-A_2) - x_1 A_2)$

$(A_2 + x_2 G)\ (x_1)$

Handling Multiplication [BGG+14]

Recall $G\,G^{-1}(A) = A$

$C_1 = (A_1 + x_1 G)^T s + \text{noise}$ $\qquad$ $C_2 = (A_2 + x_2 G)^T s + \text{noise}$

Want $\quad C_{x1\ x2} = (A_{12} + x_1 x_2 G)^T s + \text{noise}$

Key Observation: x may be used in evaluation !

$$(A_1 + x_1 G)\ G^{-1}(-A_2) \quad = (A_1 G^{-1}(-A_2) - x_1 A_2)$$

$$(A_2 + x_2 G)\ (x_1) \quad = (x_1 A_2 + x_1 x_2 G)$$

Handling Multiplication [BGG+14]

Recall $G\,G^{-1}(A) = A$

$C_1 = (A_1 + x_1\,G)^T s + \text{noise}$ $\quad C_2 = (A_2 + x_2\,G)^T s + \text{noise}$

Want $\quad C_{x1\;x2} = (A_{12} + x_1x_2\,G)^T s + \text{noise}$

Key Observation: x may be used in evaluation !

$$(A_1 + x_1\,G)\;G^{-1}(-A_2) \quad = (A_1\,G^{-1}(-A_2) - x_1\,A_2)$$

$$+$$

$$(A_2 + x_2\,G)\,(x_1) \quad = (x_1\,A_2 + x_1x_2\,G)$$

Handling Multiplication [BGG+14]

Recall $\boxed{G\,G^{-1}(A) = A}$

$C_1 = (A_1 + x_1 G)^T s + \text{noise}$ $\qquad$ $C_2 = (A_2 + x_2 G)^T s + \text{noise}$

Want $\quad C_{x1\ x2} = (A_{12} + x_1 x_2 G)^T s + \text{noise}$

Key Observation: x may be used in evaluation !

$$
\begin{array}{lll}
 & (A_1 + x_1 G)\ G^{-1}(-A_2) & = (A_1 G^{-1}(-A_2) - \cancel{x_1 A_2}) \\
+ & & \\
 & (A_2 + x_2 G)\ (x_1) & = (\cancel{x_1 A_2} + x_1 x_2 G)
\end{array}
$$

Handling Multiplication [BGG+14]

Recall $G\,G^{-1}(A) = A$

$C_1 = (A_1 + x_1\,G)^T\,s + \text{noise}$ $\qquad$ $C_2 = (A_2 + x_2\,G)^T\,s + \text{noise}$

Want $\quad C_{x1\;x2} = (A_{12} + x_1x_2\,G)^T\,s + \text{noise}$

Key Observation: x may be used in evaluation !

$$
\begin{aligned}
& (A_1 + x_1\,G)\;G^{-1}(-A_2) && = (A_1\,G^{-1}(-A_2) - \cancel{x_1A_2}) \\
+\; & (A_2 + x_2\,G)\;(x_1) && = (\cancel{x_1A_2} + x_1x_2\,G) \\
& = (A_{12} + x_1x_2\,G) &&
\end{aligned}
$$

Handling Multiplication [BGG+14]

Handling Multiplication [BGG+14]

$C_1 = (A_1 + x_1 G)^T s + \text{noise}$ $\quad$ $C_2 = (A_2 + x_2 G)^T s + \text{noise}$

Handling Multiplication [BGG+14]

Let $\boxed{R = G^{-1}(-A_2)}$

$C_1 = (A_1 + x_1 G)^T s + \text{noise}$ $\quad$ $C_2 = (A_2 + x_2 G)^T s + \text{noise}$

Handling Multiplication [BGG+14]

Let $\boxed{R = G^{-1}(-A_2)}$

$C_1 = (A_1 + x_1 G)^T s + \text{noise}$ $C_2 = (A_2 + x_2 G)^T s + \text{noise}$

Then $C_{x1\ x2} = R^T C_1 + x_1 C_2$

$= (A_{12} + x_1 x_2 G)^T s + \text{noise}$

$A_{12} = A_1 G^{-1}(-A_2)$

Handling Multiplication [BGG+14]

Let $R = G^{-1}(-A_2)$

$C_1 = (A_1 + x_1 G)^T s + \text{noise}$ $\quad$ $C_2 = (A_2 + x_2 G)^T s + \text{noise}$

Then $C_{x1\ x2} = R^T C_1 + x_1 C_2$

$$= (A_{12} + x_1 x_2 G)^T s + \text{noise}$$

$$A_{12} = A_1 G^{-1}(-A_2)$$

$G^{-1}(-A_2)$ and x_1 are small and do not affect noise !

Handling Multiplication [BGG+14]

Let $R = G^{-1}(-A_2)$

$C_1 = (A_1 + x_1 G)^T s + \text{noise}$ $\quad$ $C_2 = (A_2 + x_2 G)^T s + \text{noise}$

Then $C_{x1\ x2} = R^T C_1 + x_1 C_2$

$= (A_{12} + x_1 x_2 G)^T s + \text{noise}$

$A_{12} = A_1 G^{-1}(-A_2)$

$G^{-1}(-A_2)$ and x_1 are small and do not affect noise !

Also have $C = u^T s + \text{noise} + \text{msg}$, $C' = A^T s + \text{noise}$

Handling Multiplication [BGG+14]

Let $R = G^{-1}(-A_2)$

$C_1 = (A_1 + x_1 G)^T s + \text{noise}$ $C_2 = (A_2 + x_2 G)^T s + \text{noise}$

Then $C_{x1\ x2} = R^T C_1 + x_1 C_2$

$= (A_{12} + x_1 x_2 G)^T s + \text{noise}$

$A_{12} = A_1 G^{-1}(-A_2)$

$G^{-1}(-A_2)$ and x_1 are small and do not affect noise !

Also have $C = u^T s + \text{noise} + \text{msg}$, $C' = A^T s + \text{noise}$

If $x_1 x_2 = 0$, then $C' \mid C_{x1\ x2} = [A \mid A_{12}]^T s + \text{noise}$

Handling Multiplication [BGG+14]

Handling Multiplication [BGG+14]

If $x_1 x_2 = 0$, then $C' \mid C_{x1\ x2} = [A \mid A_{12}]^T s$ + noise

Handling Multiplication [BGG+14]

If $x_1 x_2 = 0$, then $C' \mid C_{x1\ x2} = [A \mid A_{12}]^T s + \text{noise}$

$$\text{Key} \quad \begin{bmatrix} A & A_{12} \end{bmatrix} \begin{bmatrix} e_{12} \end{bmatrix} \equiv \begin{bmatrix} u \end{bmatrix} \mod q$$

Handling Multiplication [BGG+14]

If $x_1 x_2 = 0$, then $C' \mid C_{x1\ x2} = [A \mid A_{12}]^T s + \text{noise}$

$$\text{Key} \quad \begin{bmatrix} A & A_{12} \end{bmatrix} \begin{bmatrix} e_{12} \end{bmatrix} \equiv \begin{bmatrix} u \end{bmatrix} \mod q$$

Perform Regev Decryption

Handling Multiplication [BGG+14]

If $x_1 x_2 = 0$, then $C' \mid C_{x1\ x2} = [A \mid A_{12}]^T s + \text{noise}$

Key $\quad [\,A \mid A_{12}\,]\ e_{12} \equiv u \mod q$

Perform Regev Decryption

$(e_{12})^T [C' \mid C_{x1\ x2}] = (e_{12})^T [A \mid A_{12}]^T s + (e_{12})^T \text{noise} = u^T s + \text{noise}$

Handling Multiplication [BGG+14]

If $x_1x_2 = 0$, then $C' \mid C_{x1\ x2} = [A \mid A_{12}]^T s + \text{noise}$

Key $\{A \mid A_{12}\}\ e_{12} \equiv u \mod q$

Perform Regev Decryption

$C = u^T s + \text{noise} + \text{msg}$

$(e_{12})^T [C' \mid C_{x1\ x2}] = (e_{12})^T [A \mid A_{12}]^T s + (e_{12})^T \text{noise} = u^T s + \text{noise}$

Handling Multiplication [BGG+14]

If $x_1 x_2 = 0$, then $C' \mid C_{x1\ x2} = [A \mid A_{12}]^T s + \text{noise}$

$$\text{Key} \quad \boxed{A \;\big|\; A_{12}} \; \boxed{e_{12}} \;\equiv\; \boxed{u} \quad \text{mod } q$$

Perform Regev Decryption

$$C = u^T s + \text{noise} + \text{msg}$$

$-$

$$(e_{12})^T [C' \mid C_{x1\ x2}] = (e_{12})^T [A \mid A_{12}]^T s + (e_{12})^T \text{noise} = u^T s + \text{noise}$$

Handling Multiplication [BGG+14]

If $x_1 x_2 = 0$, then $C' \mid C_{x1\ x2} = [A \mid A_{12}]^T s + \text{noise}$

$$\text{Key} \quad \begin{bmatrix} A & A_{12} \end{bmatrix} \begin{bmatrix} e_{12} \end{bmatrix} \equiv \begin{bmatrix} u \end{bmatrix} \mod q$$

Perform Regev Decryption

$$C \;\; = u^T s + \text{noise} + \text{msg}$$

$$- \;(e_{12})^T [C' \mid C_{x1\ x2}] = (e_{12})^T [A \mid A_{12}]^T s + (e_{12})^T \text{noise} = u^T s + \text{noise}$$

$$= \text{noise} + \text{msg}$$

More Generally [BGG+14]. . .

More Generally [BGG+14]…

There exist "small" $\widehat{\mathbf{H}}_{f,\mathbf{x}}$, $\mathbf{H}_f$ such that:

$$[\ \mathbf{A}_1 - x_1\mathbf{G}\ |\ \ldots\ldots\ |\ \mathbf{A}_n - x_n\mathbf{G}\ ]\ \widehat{\mathbf{H}}_{f,\mathbf{x}} = [\ \mathbf{A}_1\ |\ \ldots\ |\ \mathbf{A}_n]\ \mathbf{H}_f - f(\mathbf{x})\ \mathbf{G}$$

More Generally [BGG+14]…

There exist “small” $\widehat{\mathbf{H}}_{f,\mathbf{x}}$, $\mathbf{H}_f$ such that:

$$[\,\mathbf{A}_1 - x_1\mathbf{G} \mid \ldots\ldots \mid \mathbf{A}_n - x_n\mathbf{G}\,]\ \widehat{\mathbf{H}}_{f,\mathbf{x}} = \underbrace{[\,\mathbf{A}_1 \mid \ldots \mid \mathbf{A}_n]\ \mathbf{H}_f}_{\mathrm{A_f}} - f(\mathbf{x})\ \mathbf{G}$$

More Generally [BGG+14]…

There exist "small" $\widehat{\mathbf{H}}_{f,\mathbf{x}}$, $\mathbf{H}_f$ such that:

$$[\,\mathbf{A}_1 - x_1\mathbf{G}\,|\,\ldots\ldots\,|\,\mathbf{A}_n - x_n\mathbf{G}\,]\ \widehat{\mathbf{H}}_{f,\mathbf{x}} = \underbrace{[\,\mathbf{A}_1\,|\,\ldots\,|\,\mathbf{A}_n]\ \mathbf{H}_f}_{A_f} - f(\mathbf{x})\ \mathbf{G}$$

Recall $C_i = (A_i + x_i\ G)^T\ s + \text{noise}$

More Generally [BGG+14]...

There exist "small" $\widehat{\mathbf{H}}_{f,\mathbf{x}}$, $\mathbf{H}_f$ such that:

$$[\,\mathbf{A}_1 - x_1\mathbf{G}\,|\,\ldots\ldots\,|\,\mathbf{A}_n - x_n\mathbf{G}\,]\,\widehat{\mathbf{H}}_{f,\mathbf{x}} = \underbrace{[\,\mathbf{A}_1\,|\,\ldots\,|\,\mathbf{A}_n]\,\mathbf{H}_f}_{A_f} - f(\mathbf{x})\,\mathbf{G}$$

Recall $C_i = (A_i + x_i\, G)^T s$ + noise

LHS implies that

More Generally [BGG+14]…

There exist "small" $\widehat{\mathbf{H}}_{f,\mathbf{x}}$, $\mathbf{H}_f$ such that:

$$[\ \mathbf{A}_1 - x_1\mathbf{G}\ |\ \ldots\ldots\ |\ \mathbf{A}_n - x_n\mathbf{G}\ ]\ \widehat{\mathbf{H}}_{f,\mathbf{x}} = \underbrace{[\ \mathbf{A}_1\ |\ \ldots\ |\ \mathbf{A}_n]\ \mathbf{H}_f}_{\mathrm{A_f}} - f(\mathbf{x})\ \mathbf{G}$$

Recall $\mathrm{C_i = (A_i + x_i\ G)^T\ s + noise}$

LHS implies that

$$\widehat{\mathbf{H}}^T_{f,\mathbf{x}}\ [\ \mathbf{C}_1\ |\ \ldots\ldots\ |\ \mathbf{C}_n\ ] = [\mathbf{A}_f - f(\mathbf{x})\ \mathbf{G}]^T\mathbf{s} + \textbf{noise}$$

More Generally [BGG+14]…

There exist "small" $\widehat{\mathbf{H}}_{f,\mathbf{x}}, \mathbf{H}_f$ such that:

$$[\,\mathbf{A}_1 - x_1\mathbf{G} \,|\, \ldots\ldots \,|\, \mathbf{A}_n - x_n\mathbf{G}\,]\ \widehat{\mathbf{H}}_{f,\mathbf{x}} = \underbrace{[\,\mathbf{A}_1 \,|\, \ldots \,|\, \mathbf{A}_n]\ \mathbf{H}_f}_{A_f} - f(\mathbf{x})\ \mathbf{G}$$

Recall $C_i = (A_i + x_i\ G)^T\ s$ + noise

LHS implies that

$$\widehat{\mathbf{H}}^T_{f,\mathbf{x}}\ [\,\mathbf{C}_1 \,|\, \ldots\ldots \,|\, \mathbf{C}_n\,] = [\mathbf{A}_f - f(\mathbf{x})\ \mathbf{G}]^T \mathbf{s} + \textbf{noise}$$

Keygen provides matching key

$$\begin{bmatrix} A & A_f \end{bmatrix} \begin{bmatrix} e_f \end{bmatrix} \equiv \begin{bmatrix} u \end{bmatrix} \bmod q$$

More Generally [BGG+14]…

There exist “small” $\widehat{\mathbf{H}}_{f,\mathbf{x}}, \mathbf{H}_f$ such that:

$$[\,\mathbf{A}_1 - x_1\mathbf{G} \,|\, \ldots\ldots \,|\, \mathbf{A}_n - x_n\mathbf{G}\,]\ \widehat{\mathbf{H}}_{f,\mathbf{x}} = \underbrace{[\,\mathbf{A}_1 \,|\, \ldots \,|\, \mathbf{A}_n]\ \mathbf{H}_f}_{\mathrm{A_f}} - f(\mathbf{x})\ \mathbf{G}$$

Recall $C_i = (A_i + x_i\ G)^T\ s + \text{noise}$

LHS implies that

$$\widehat{\mathbf{H}}^{T}_{f,\mathbf{x}}\ [\,\mathbf{C}_1 \,|\, \ldots\ldots \,|\, \mathbf{C}_n\,] = [\mathbf{A}_f - f(\mathbf{x})\ \mathbf{G}]^T\mathbf{s} + \textbf{noise}$$

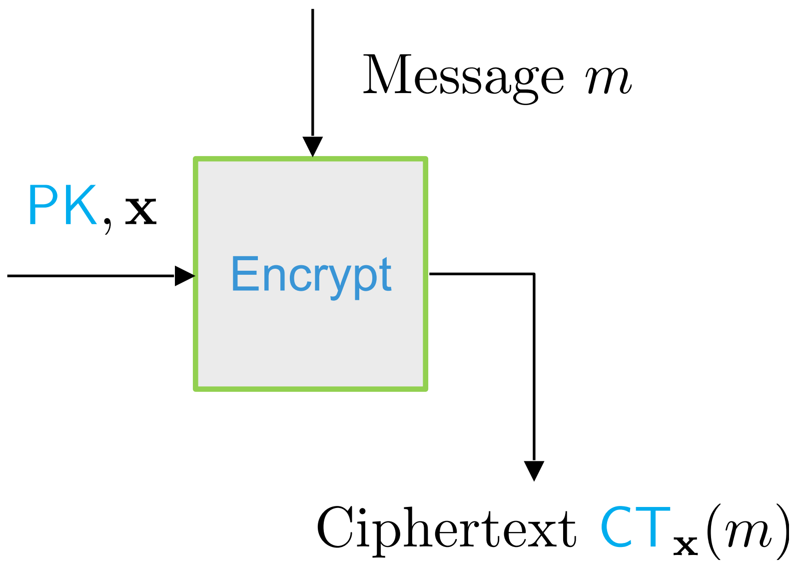
Keygen provides matching key

$$\begin{bmatrix} \mathrm{A} & \mathrm{A_f} \end{bmatrix} \begin{bmatrix} \mathrm{e_f} \end{bmatrix} \equiv \begin{bmatrix} \mathrm{u} \end{bmatrix} \mod q$$

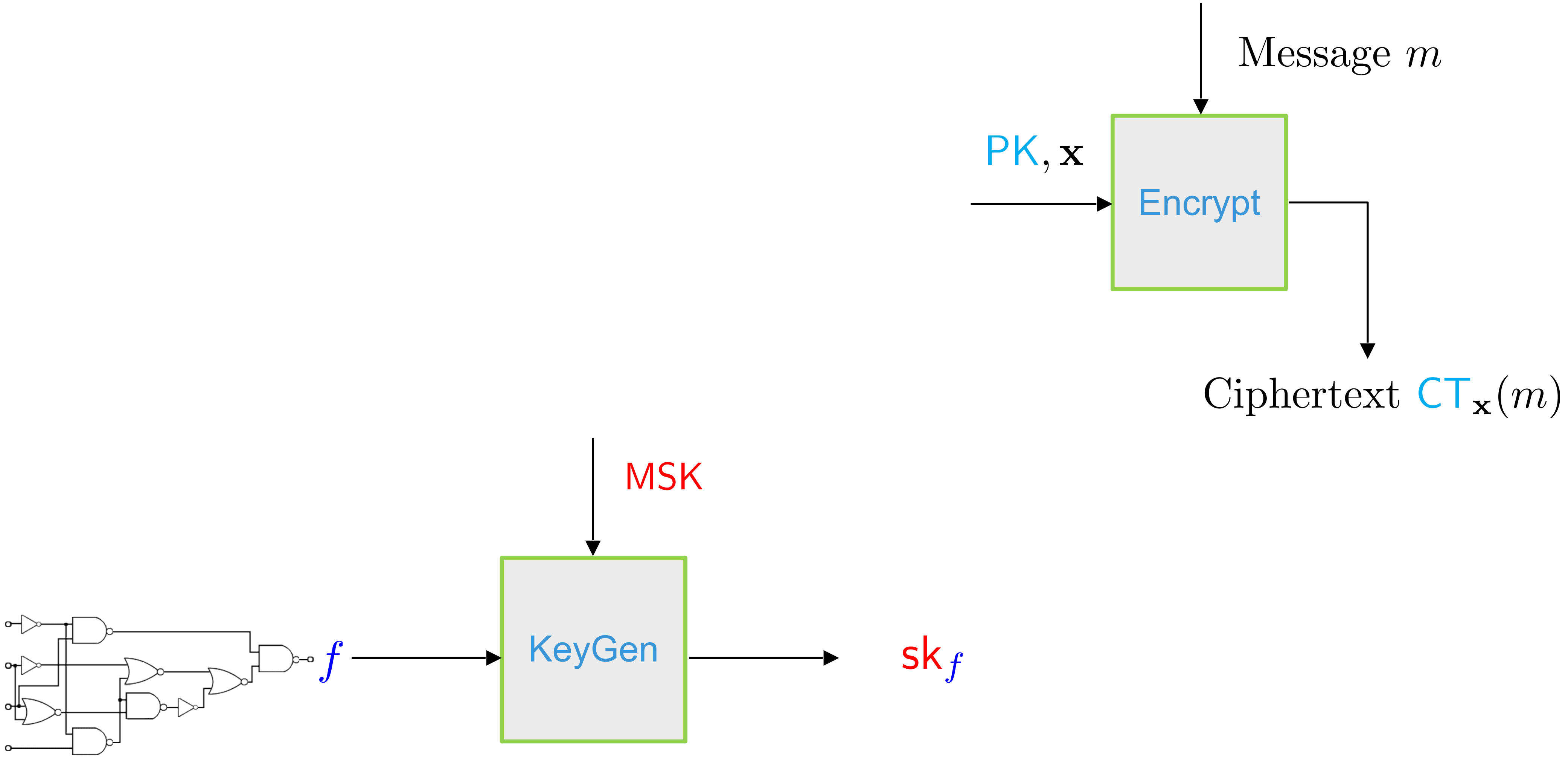
Perform Regev Decryption as usual

Generalizes to all circuits [BGG+14]

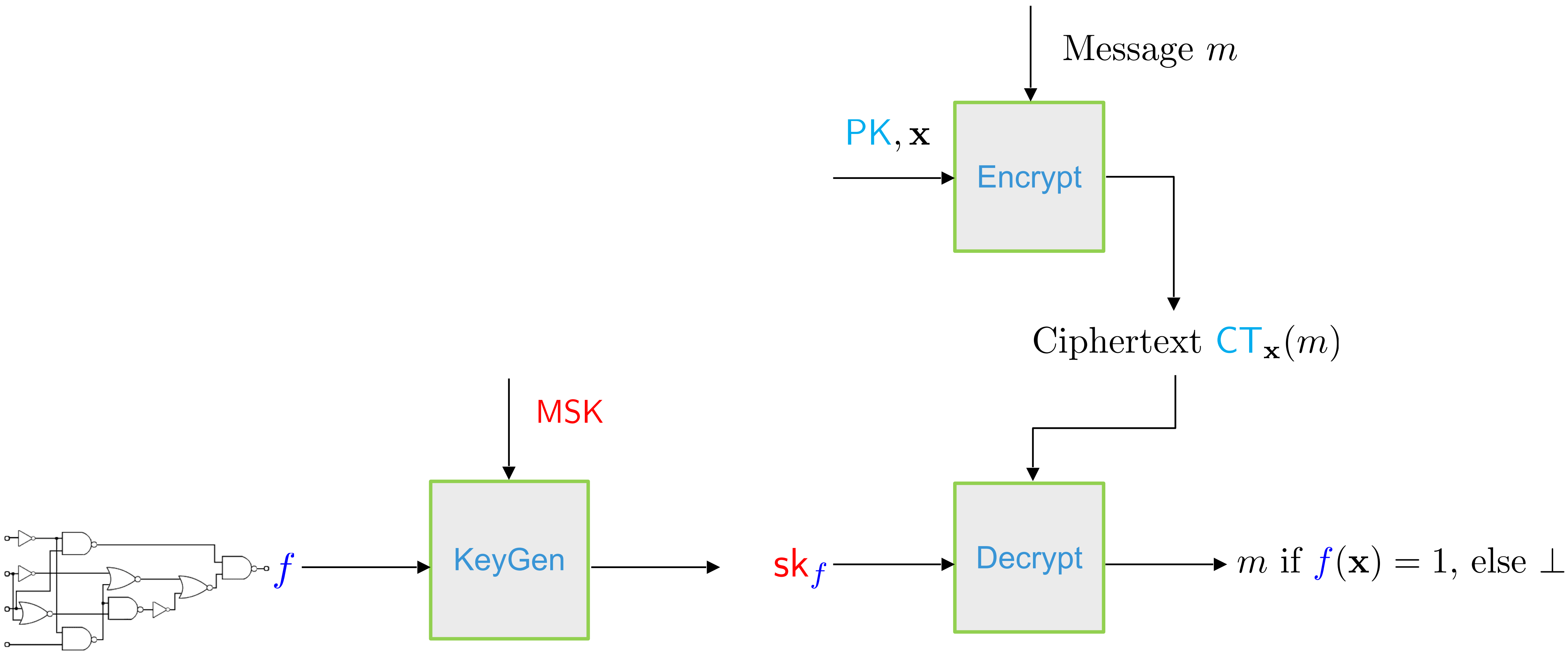
Generalizes to all circuits [BGG+14]



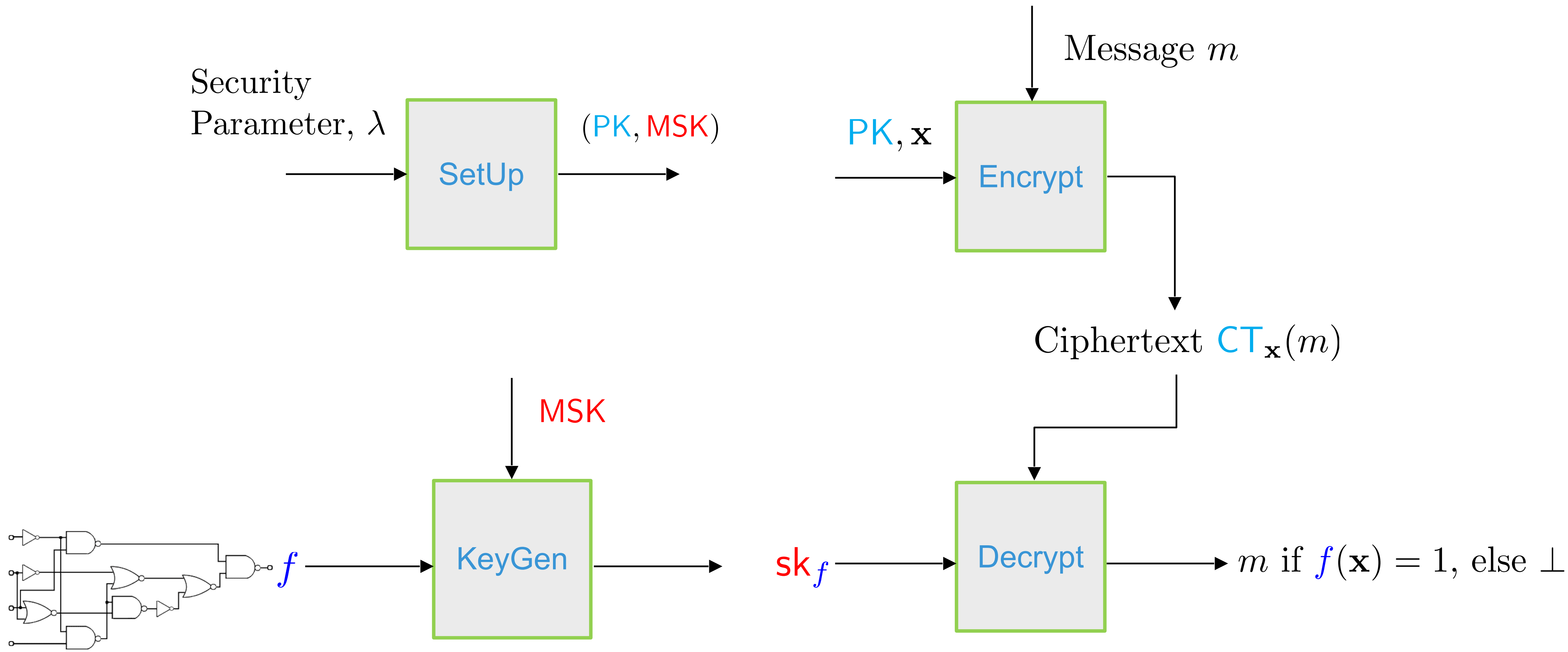
Generalizes to all circuits [BGG+14]



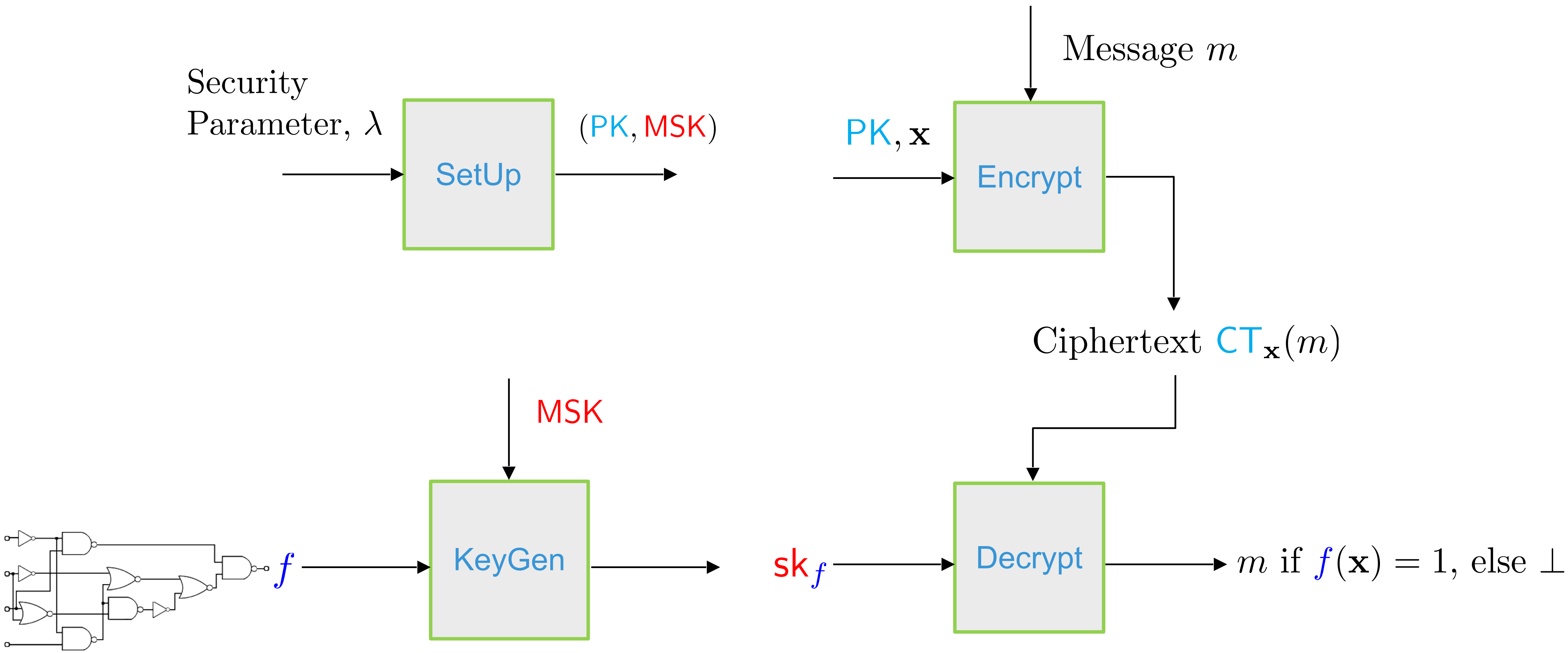
Generalizes to all circuits [BGG+14]



Generalizes to all circuits [BGG+14]

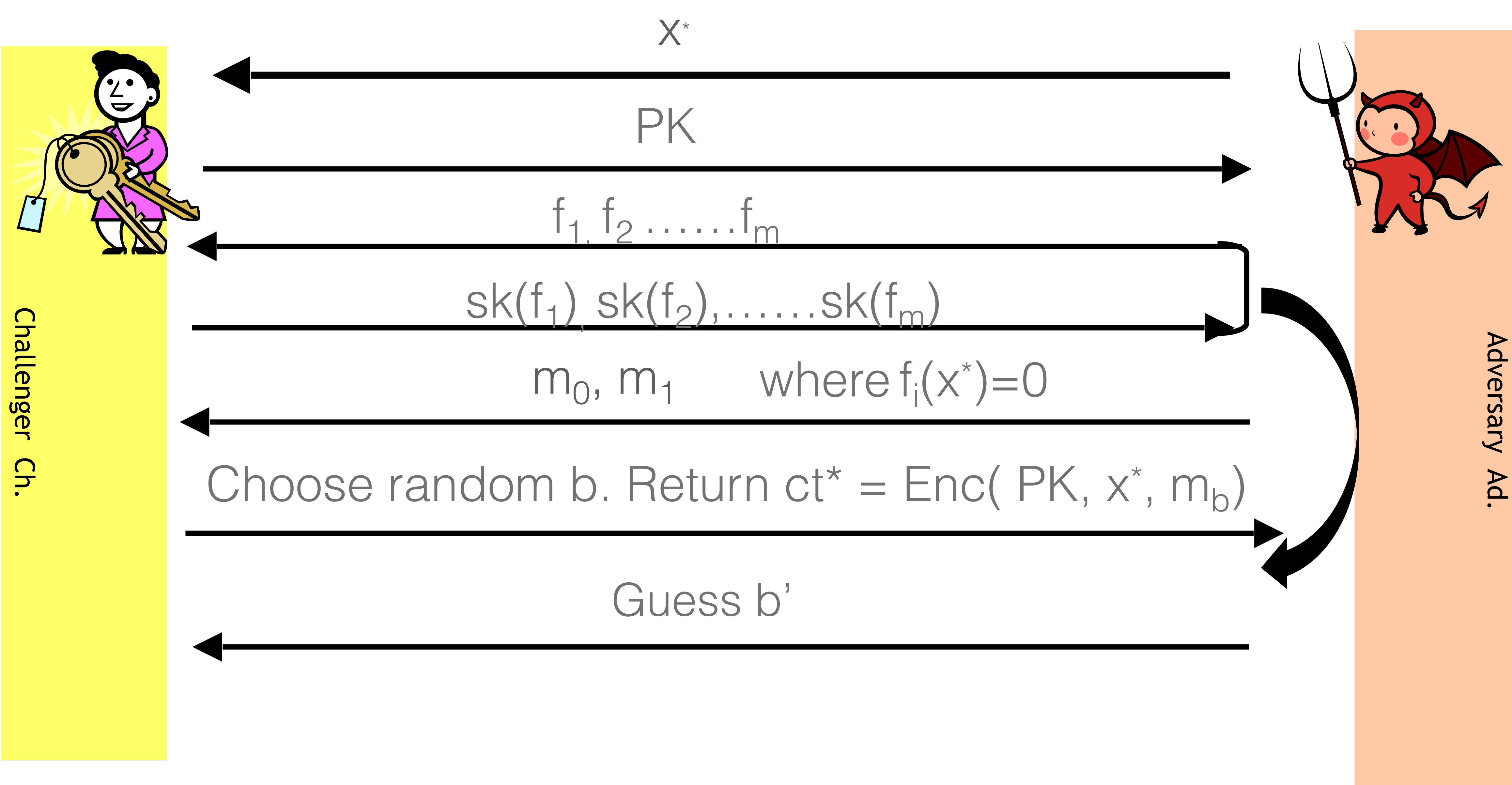


Generalizes to all circuits [BGG+14]



Attribute based Encryption (ABE) [SW05]

Security Definition



Attacker wins if | Pr[b=b'] - ½ | is non-negligible

Security: Challenges

- Challenger needs to be able to answer private key queries of Adversary: much more complex!
- Challenger can't have master trapdoor(Trapdoor for A)
- Must embed LWE challenge into challenge ciphertext

Strategy: Challenge CT

Strategy: Challenge CT

- Let x^* be challenge attributes.

Strategy: Challenge CT

- Let x^* be challenge attributes.

Strategy: Challenge CT

- Let x^* be challenge attributes.

- As before, set $A_i = [AR_i - x_i^* G]$

Strategy: Challenge CT

- Let x* be challenge attributes.
- As before, set $A_i = [AR_i - x_i^* G]$

Strategy: Challenge CT

- Let x^* be challenge attributes.

- As before, set $A_i = [AR_i - x_i^* G]$

- $C_i = (A_i + x_i G)^T s + \text{noise} = (AR_i + (x_i - x_i^*)G)^T s + \text{noise}$

Strategy: Challenge CT

- Let x^* be challenge attributes.

- As before, set $A_i = [AR_i - x_i^* G]$

- $C_i = (A_i + x_i G)^T s + \text{noise} = (AR_i + (x_i - x_i^*)G)^T s + \text{noise}$

Strategy: Challenge CT

- Let x^* be challenge attributes.
- As before, set $A_i = [AR_i - x_i^* G]$
- $C_i = (A_i + x_i G)^T s + \text{noise} = (AR_i + (x_i - x_i^*)G)^T s + \text{noise}$
- When $x = x^*$, challenge CT becomes $(AR_i)^T s + \text{noise}$

Strategy: Challenge CT

- Let x^* be challenge attributes.
- As before, set $A_i = [AR_i - x_i^* G]$
- $C_i = (A_i + x_i G)^T s + \text{noise} = (AR_i + (x_i - x_i^*)G)^T s + \text{noise}$
- When $x = x^*$, challenge CT becomes $(AR_i)^T s + \text{noise}$

Strategy: Challenge CT

- Let x^* be challenge attributes.
- As before, set $A_i = [AR_i - x_i^* G]$
- $C_i = (A_i + x_i G)^T s + \text{noise} = (AR_i + (x_i - x_i^*)G)^T s + \text{noise}$
- When $x = x^*$, challenge CT becomes $(AR_i)^T s + \text{noise}$
- Can be computed from LWE challenge

Strategy: Challenge CT

- Let x^* be challenge attributes.
- As before, set $A_i = [AR_i - x_i^* G]$
- $C_i = (A_i + x_i G)^T s + \text{noise} = (AR_i + (x_i - x_i^*)G)^T s + \text{noise}$
- When $x = x^*$, challenge CT becomes $(AR_i)^T s + \text{noise}$
- Can be computed from LWE challenge

Strategy: Key Queries

Strategy: Key Queries

- Let x^* be challenge attributes, set $A_i = [AR_i - x_i^* G]$

Strategy: Key Queries

- Let x^* be challenge attributes, set $A_i = [AR_i - x_i^* G]$

Strategy: Key Queries

- Let x^* be challenge attributes, set $A_i = [AR_i - x_i^* G]$
- Can show $A_f = [AR_f - f(x^*)G]$ for "small" R_f

Strategy: Key Queries

- Let x^* be challenge attributes, set $A_i = [AR_i - x_i^* G]$
- Can show $A_f = [AR_f - f(x^*)G]$ for "small" R_f

Strategy: Key Queries

- Let x* be challenge attributes, set $A_i = [AR_i - x_i^* G]$
- Can show $A_f = [AR_f - f(x^*)G]$ for "small" R_f
- Recall key $\begin{bmatrix} A & A_f \end{bmatrix} e_f \equiv u \mod q$

Strategy: Key Queries

- Let x^* be challenge attributes, set $A_i = [AR_i - x_i^* G]$
- Can show $A_f = [AR_f - f(x^*)G]$ for "small" R_f
- Recall key $\quad [A \mid A_f] \, e_f \equiv u \mod q$

Strategy: Key Queries

- Let x* be challenge attributes, set $A_i = [AR_i - x_i^* G]$
- Can show $A_f = [AR_f - f(x^*)G]$ for "small" R_f
- Recall key $[A \mid A_f] \, e_f \equiv u \mod q$

Strategy: Key Queries

- Let x* be challenge attributes, set $A_i = [AR_i - x_i^* G]$
- Can show $A_f = [AR_f - f(x^*)G]$ for "small" R_f
- Recall key $\begin{bmatrix} A & A_f \end{bmatrix} \begin{bmatrix} e_f \end{bmatrix} \equiv \begin{bmatrix} u \end{bmatrix} \mod q$
- Need TD for $[A \mid A_f]$ when $f(x^*)$ not 0.

Strategy: Key Queries

- Let x^* be challenge attributes, set $A_i = [AR_i - x_i^* G]$
- Can show $A_f = [AR_f - f(x^*)G]$ for "small" R_f
- Recall key $\begin{bmatrix} A & A_f \end{bmatrix} e_f \equiv u \mod q$
- Need TD for $[A \mid A_f]$ when $f(x^*)$ not 0.

Strategy: Key Queries

- Let x* be challenge attributes, set $A_i = [AR_i - x_i^* G]$
- Can show $A_f = [AR_f - f(x^*)G]$ for "small" R_f
- Recall key $$[A \mid A_f]\, e_f \equiv u \mod q$$
- Need TD for $[A \mid A_f]$ when $f(x^*)$ not 0.
- Follows from MP12

Strategy: Key Queries

- Let x* be challenge attributes, set $A_i = [AR_i - x_i^* G]$
- Can show $A_f = [AR_f - f(x^*)G]$ for "small" R_f
- Recall key $\begin{bmatrix} A & A_f \end{bmatrix} \begin{bmatrix} e_f \end{bmatrix} \equiv \begin{bmatrix} u \end{bmatrix} \bmod q$
- Need TD for $[A \mid A_f]$ when $f(x^*)$ not 0.
- Follows from MP12

Strategy: Key Queries

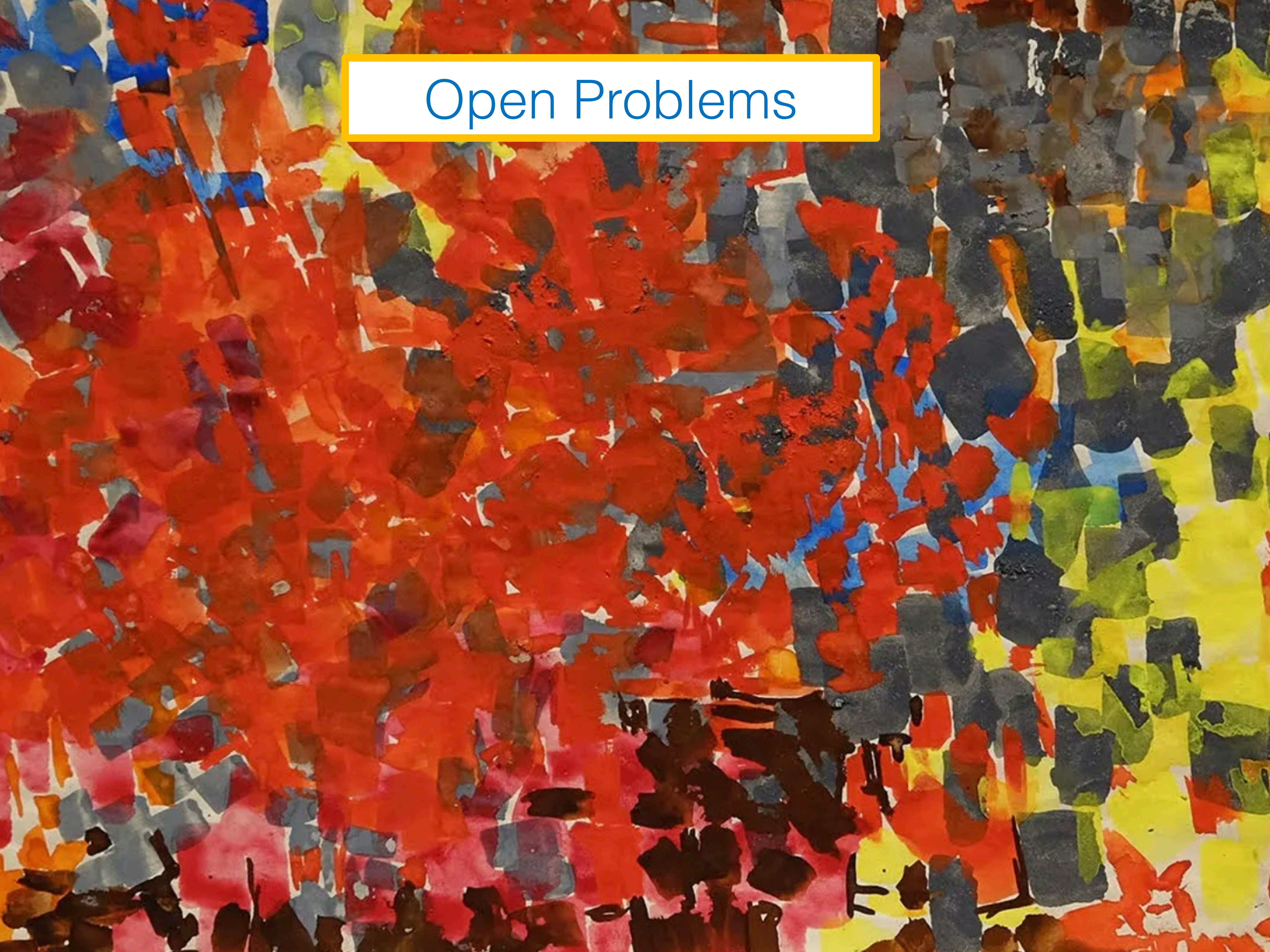
- Need TD for $[A \mid A_f]$ when $f(x^*) \neq 0$.
- $A_f = [AR_f - f(x^*)G]$. Let $H = f(x^*)$.
- Recall

Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m'}$ uniform, $\mathbf{R} \in \mathbb{Z}_q^{m' \times n \log q}$ small

Then $[\, A \mid AR - H\, G \,]$

admits LWE and SIS inversion.

Open Problems



Open Problems

- Ciphertext Policy ABE from LWE

Open Problems

- Ciphertext Policy ABE from LWE
- Broadcast Encryption from LWE

Open Problems

- Ciphertext Policy ABE from LWE
- Broadcast Encryption from LWE
- Better parameters

Open Problems

- Ciphertext Policy ABE from LWE
- Broadcast Encryption from LWE
- Better parameters
- Support uniform models of computation from LWE

Open Problems

- Ciphertext Policy ABE from LWE
- Broadcast Encryption from LWE
- Better parameters
- Support uniform models of computation from LWE
- Adaptive Security

Open Problems

- Ciphertext Policy ABE from LWE
- Broadcast Encryption from LWE
- Better parameters
- Support uniform models of computation from LWE
- Adaptive Security

Thank You!

Image Credits : Hans Hoffman, Joan Mitchell, Lynn Drexler