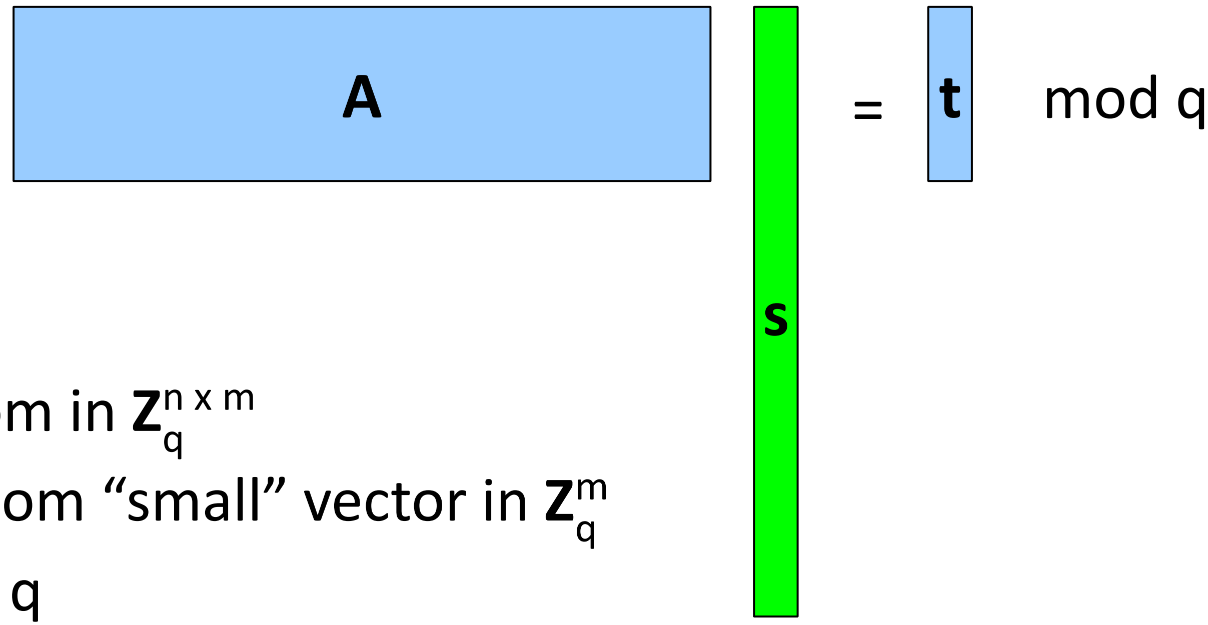
Lattice Signature Schemes

Vadim Lyubashevsky
INRIA / ENS Paris

LATTICE PROBLEMS

The Knapsack Problem



A is random in $\mathbf{Z}_q^{n \times m}$

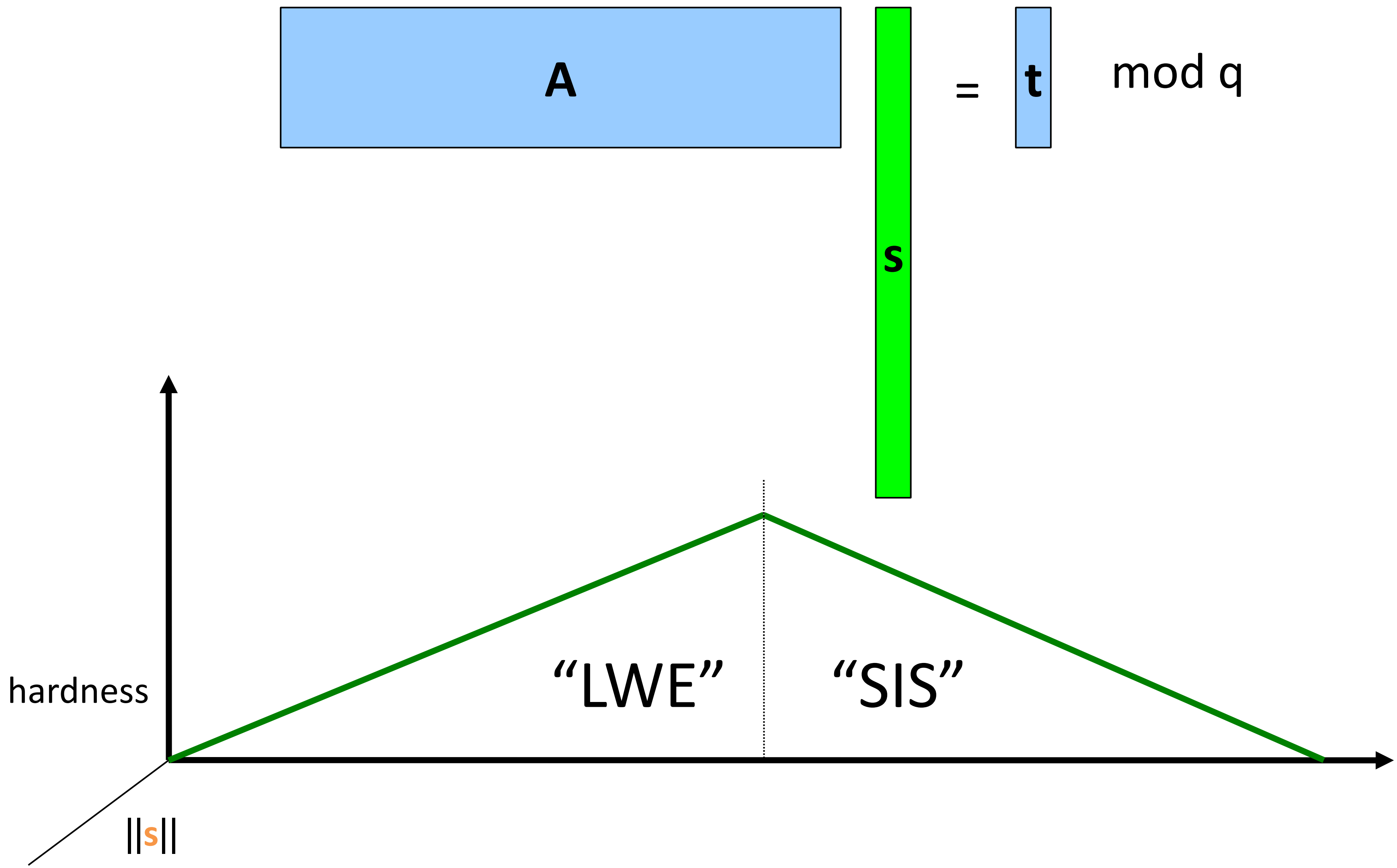
s is a random “small” vector in $\mathbf{Z}_q^{m}$

t=**A**s mod q

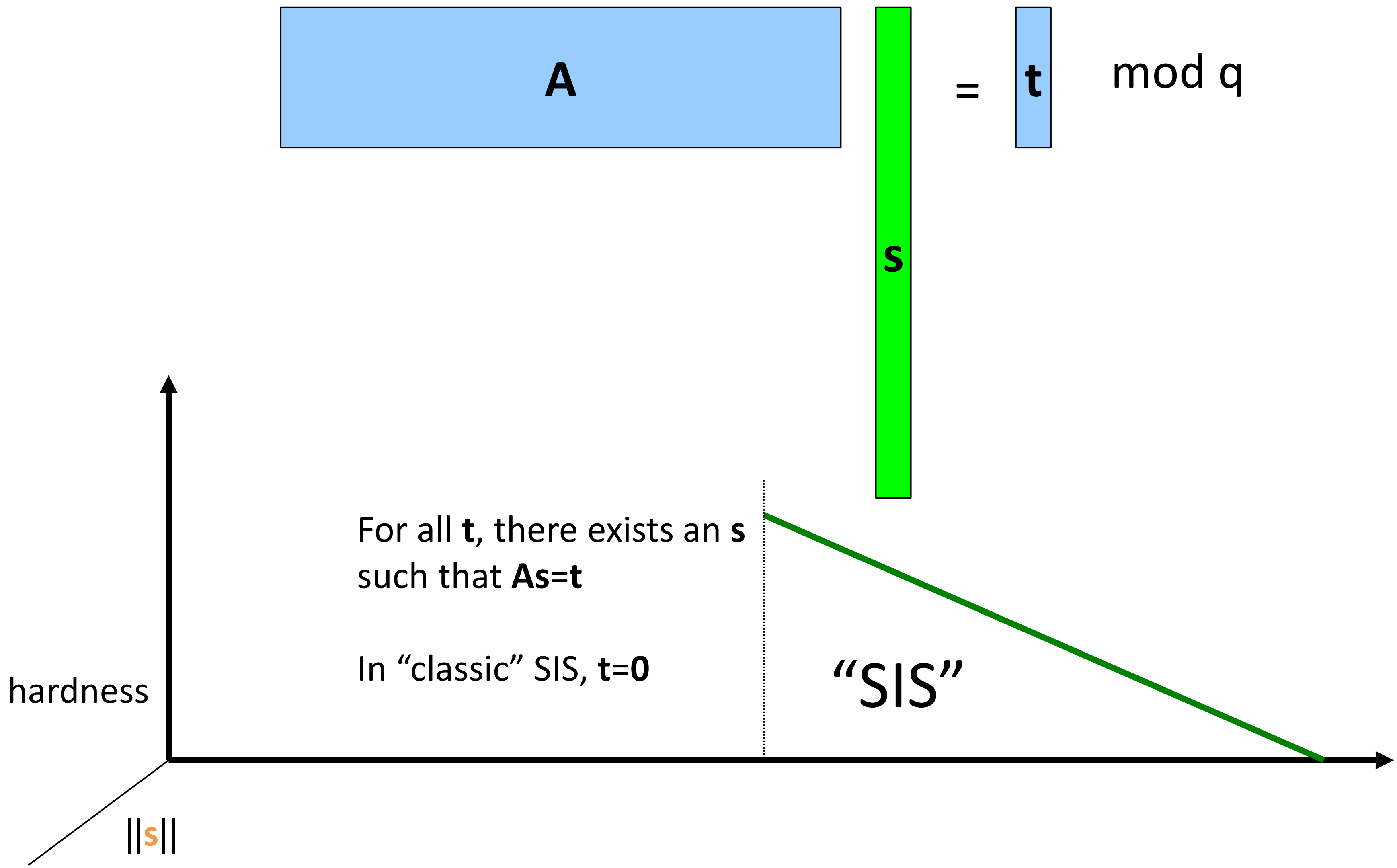
Given (**A**,**t**), find small s' such that

As'=**t** mod q

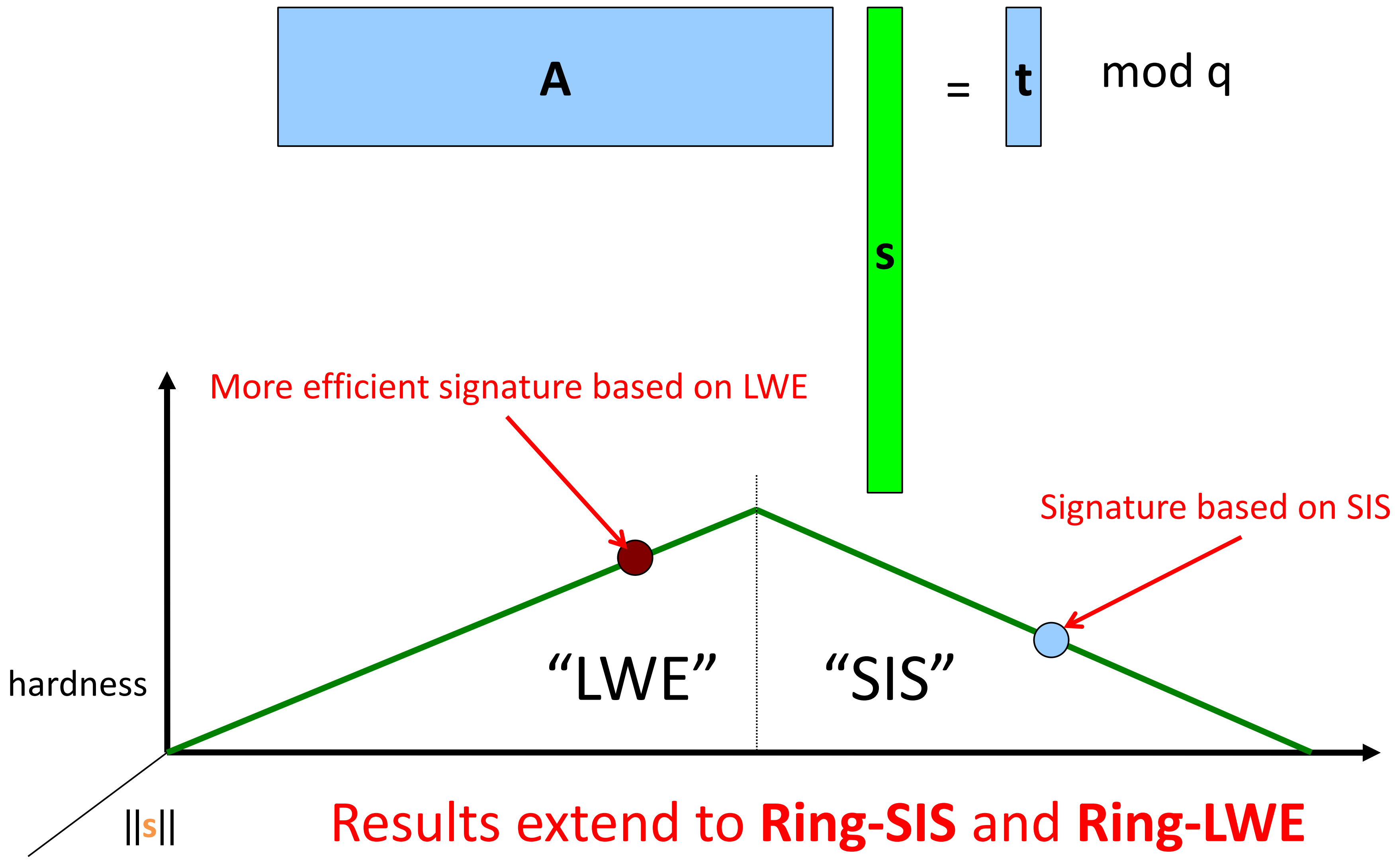
Hardness of the Knapsack Problem



Hardness of the Knapsack Problem



Results



Results extend to **Ring-SIS** and **Ring-LWE**

DIGITAL SIGNATURE SCHEMES

Digital Signatures

(sk,pk) $\leftarrow$KeyGen
Sign(sk,m_i) = s_i
Verify(pk,m_i,s_i) = YES / NO

Correctness: Verify(pk, m_i, Sign(sk,m_i)) = YES
Security: Unforgeability

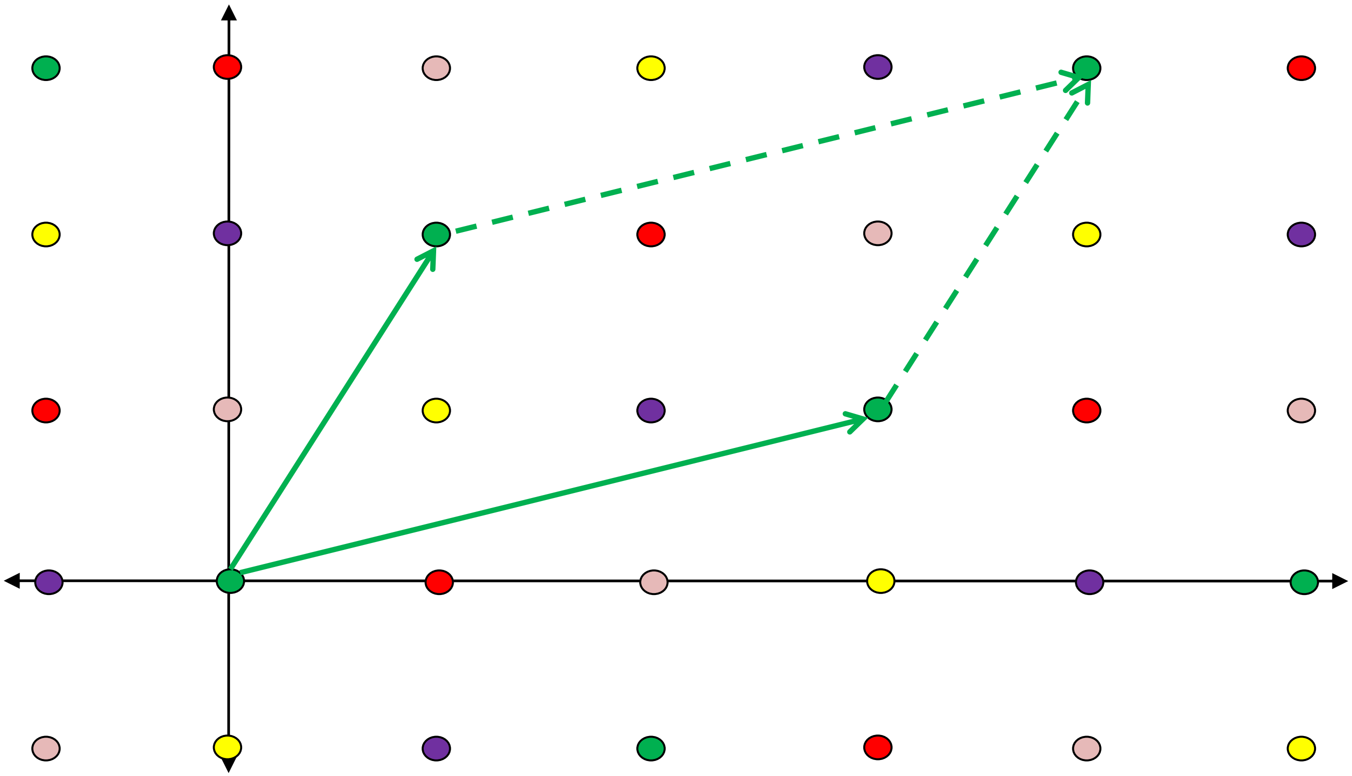
1. Adversary gets pk
2. Adversary asks for signatures of m_1, m_2, …
3. Adversary outputs (m,s) where $m \neq m_i$ and wins if Verify(pk,m,s) = YES

Signature Schemes

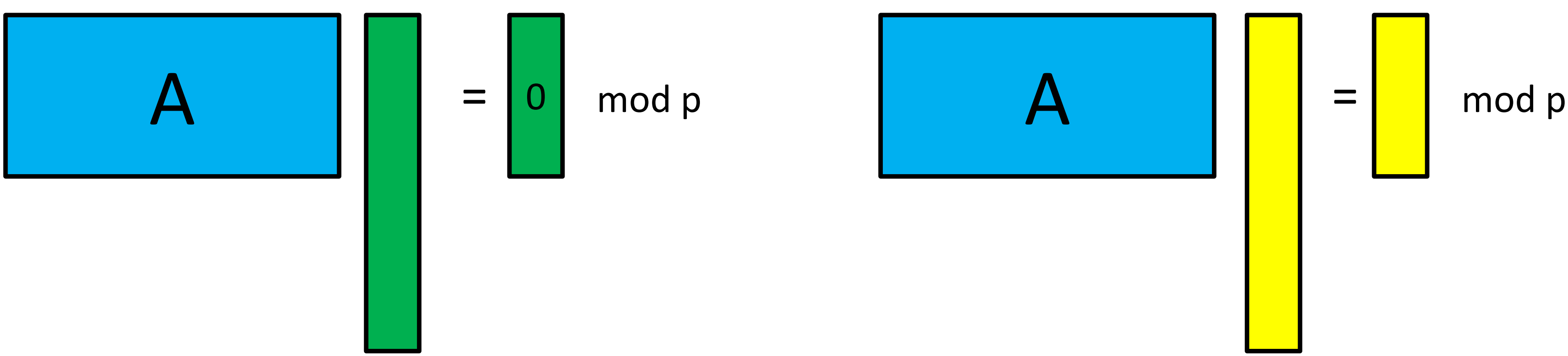
- Hash-and-Sign
  - Requires a trap-door function
- Fiat-Shamir transformation
  - Conversion from an identification scheme
  - No trap-door function needed

HASH-AND-SIGN SIGNATURE SCHEMES BASED ON SIS [GPV 2008]

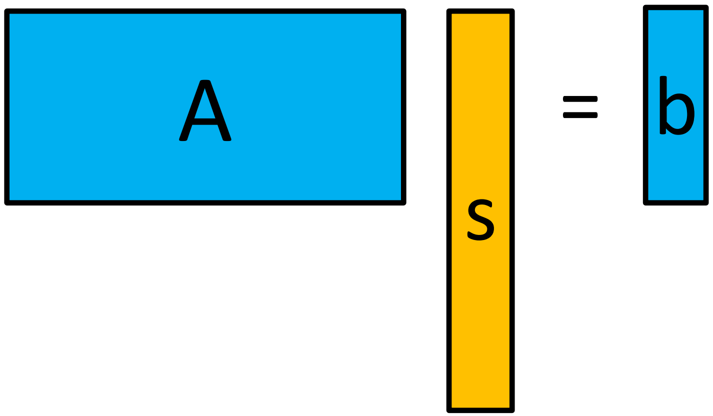
Lattice $L^{\perp}_{p}(\mathbf{A}) = \{ \mathbf{y} : \mathbf{A}\mathbf{y} = \mathbf{0} \bmod p\}$



Cosets of $\mathbf{Z}^m / L^{\perp}_{p}(\mathbf{A})$



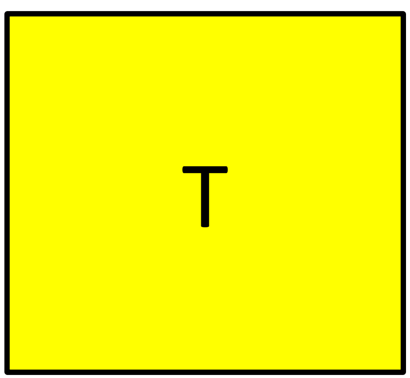
GPV Sampling



For any **b**, it outputs a short **s** such that **As**=**b** mod p

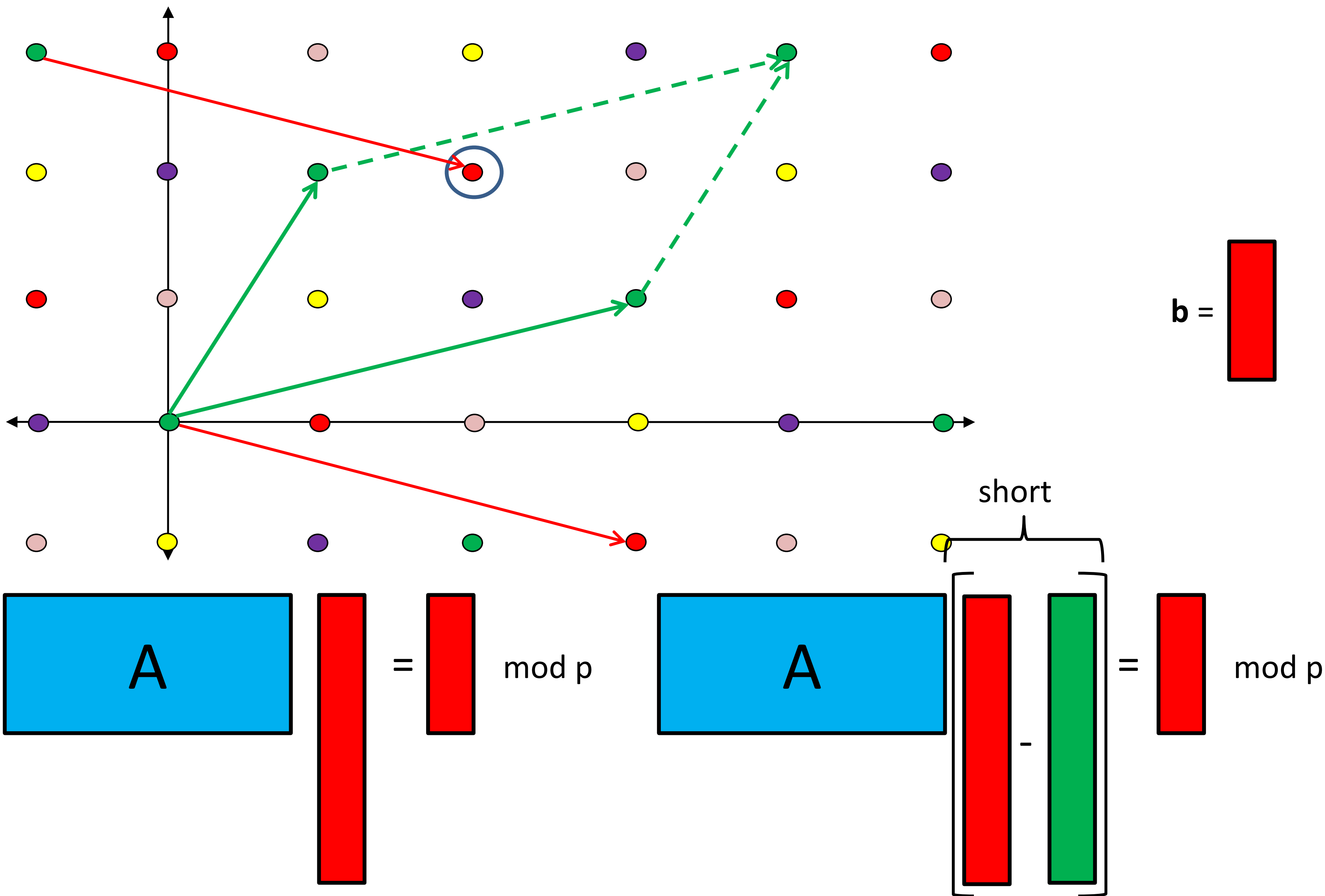
Distribution D of **s** only depends on the length of the vectors comprising **T**

T is a basis for $L_p^{\perp}(\mathbf{A})$ and has “short” vectors

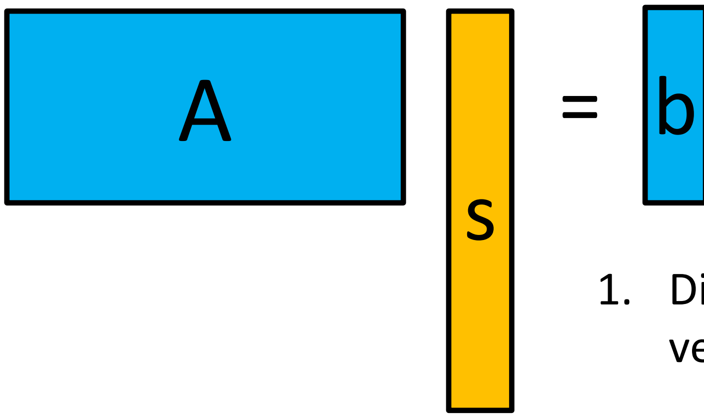


Lattice $L_p^\perp(\mathbf{A}) = \{\, \mathbf{y} : \mathbf{A}\mathbf{y} = \mathbf{0} \bmod p \,\}$

GPV Sampling



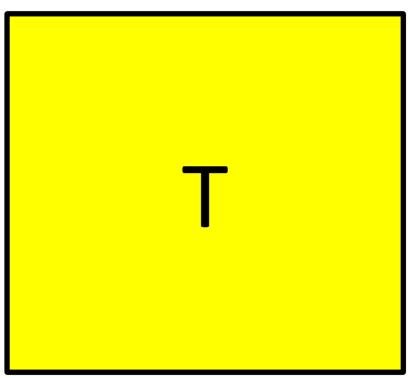
Properties Needed



1. Distribution D of **s** only depends on the length of the vectors comprising **T**

2. The following produce the same distribution of (**s**,**b**)

   (a) Choose **s** ~ D. Set **b**=**As**
   (b) Choose random **b**. Use **T** to find an **s** such that **As**=**b**.



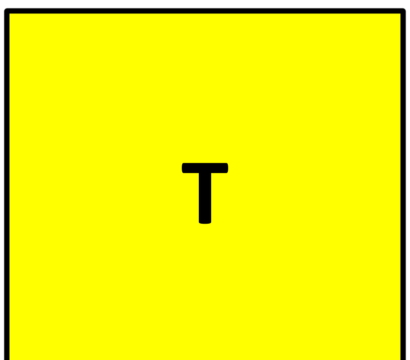
(1) is guaranteed by the GPV algorithm
(2) is true if **s** has enough entropy (to make **As**=**b** uniform mod p)

Hash-and-Sign Lattice Signature

$\mathbf{A}\,\mathbf{s} = \mathbf{b}$

Public Key: **A**
Secret Key: **T**

Lattice $L_p^{\perp}(\mathbf{A}) = \{\, \mathbf{y} : \mathbf{A}\mathbf{y} = \mathbf{0} \bmod p \,\}$

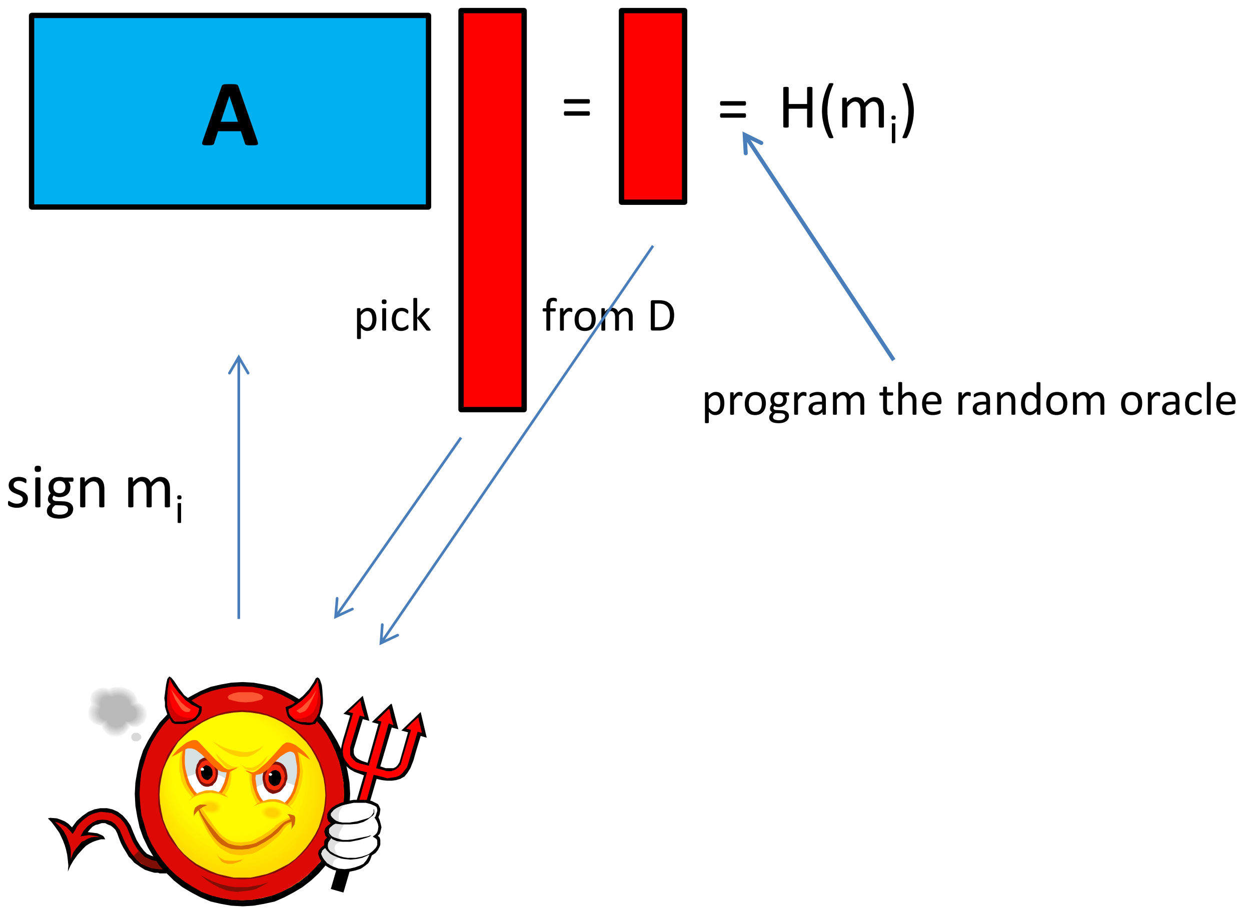


T is a basis for $L_p^{\perp}(\mathbf{A})$ and has "short" vectors

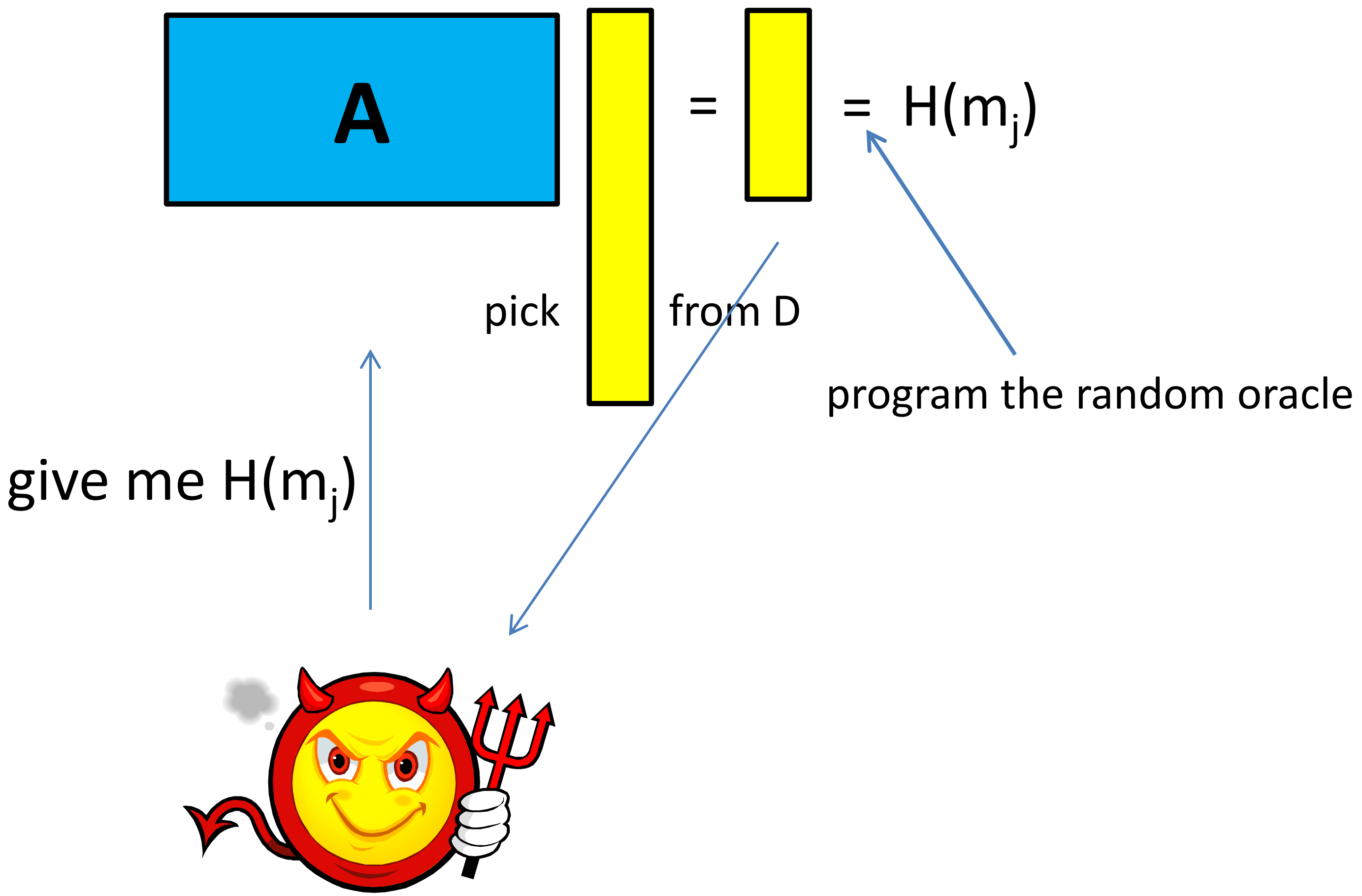
Sign(**T**,m)
1. **b** = H(m)
2. Use the GPV algorithm to find a short **s** such that **As** = **b** mod p
3. **s** is the signature of m

Verify(**A**,m,**s**)
1. check that **s** is "short" and
   As = H(m) mod p

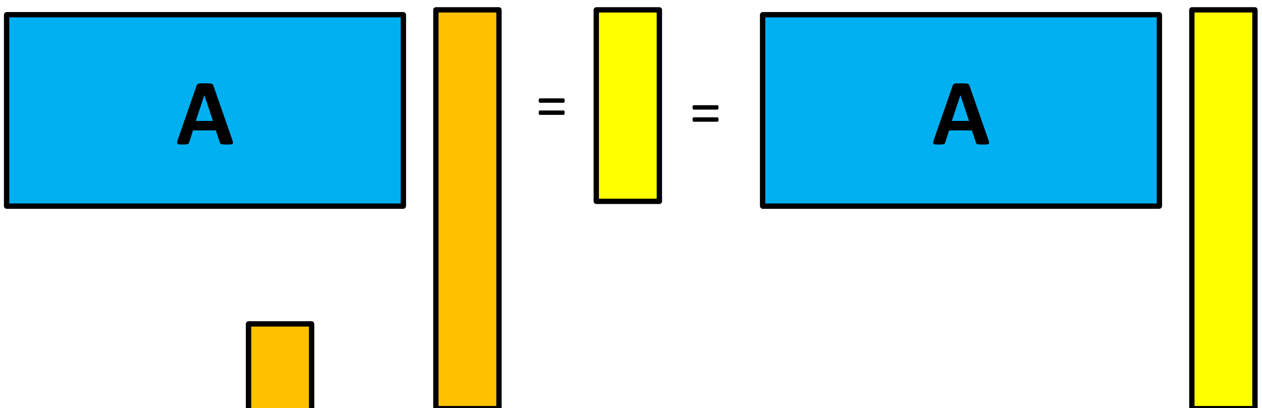
Security Proof Sketch



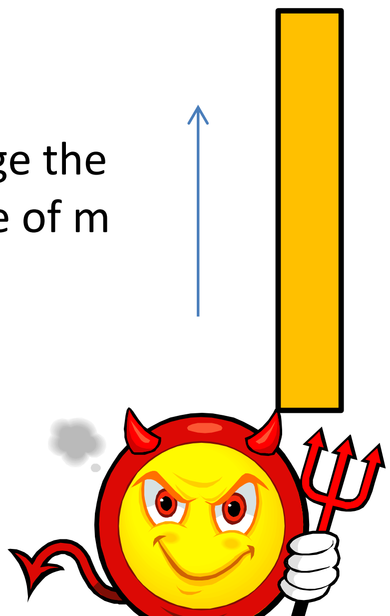
Security Proof Sketch



Security Proof Sketch



I will forge the signature of m

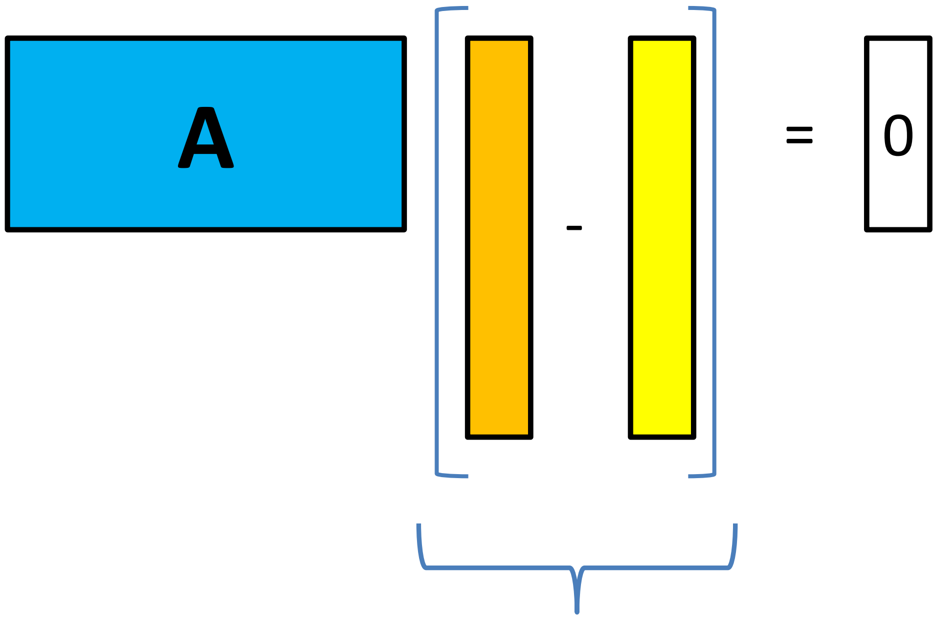


To forge on m, the Adversary needs H(m)

So m is one of the m_j he asked for $H(m_j)$

Thus we know an s_j such that $As_j=H(m_j)$

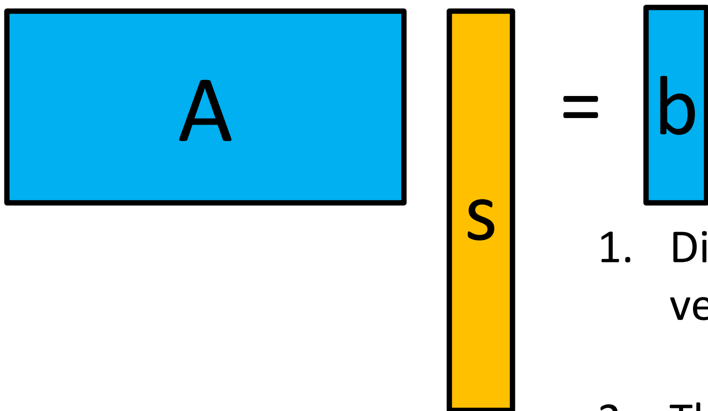
Security Proof Sketch



short and hopefully non-zero

if it’s non-zero, then we have a solution to SIS

Properties Needed

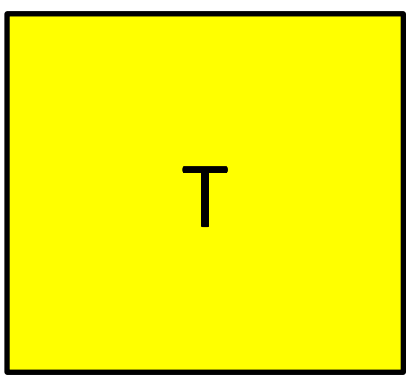


1. Distribution D of **s** only depends on the length of the vectors comprising **T**

2. The following produce the same distribution of (**s**,**b**)

   (a) Choose **s** ~ D. Set **b**=**As**
   (b) Choose random **b**. Use **T** to find an **s** such that **As**=**b**.

3. For a random **b**, there is more than one likely possible output **s** such that **b**=**As**.



(1) is guaranteed by the GPV algorithm
(2) is true if **s** has enough entropy (to make A**s**=**b** uniform mod p)
(3) is true because the standard deviation of GPV is big

FIAT-SHAMIR SIGNATURE SCHEMES BASED ON SIS
[L '09, L'12, DDLL '13]

Signature Scheme (Main Idea)

Secret Key: $\mathbf{S}$

Public Key: $\mathbf{A}$, $\mathbf{T}=\mathbf{A}\mathbf{S} \bmod q$

<u>Sign</u>(μ)

Pick a random $\mathbf{y}$

Compute $\mathbf{c}=H(\mathbf{A}\mathbf{y} \bmod q, \mu)$

$\mathbf{z}=\mathbf{S}\mathbf{c}+\mathbf{y}$

Output($\mathbf{z}$,$\mathbf{c}$)

<u>Verify</u>($\mathbf{z}$,$\mathbf{c}$)

Check that $\mathbf{z}$ is "small"

and

$\mathbf{c} = H(\mathbf{A}\mathbf{z} - \mathbf{T}\mathbf{c} \bmod q, \mu)$

Security Reduction Requirements

Secret Key: $\mathbf{S}$

Given the public key, the secret key is not unique

Public Key: $\mathbf{A}$, $\mathbf{T}=\mathbf{A}\mathbf{S} \bmod q$

<u>Sign</u>(μ)

Pick a random $\mathbf{y}$

Compute $\mathbf{c}=H(\mathbf{A}\mathbf{y} \bmod q, \mu)$

$\mathbf{z}=\mathbf{S}\mathbf{c}+\mathbf{y}$

Output($\mathbf{z}$,$\mathbf{c}$)

Signature is independent of the secret key

<u>Verify</u>($\mathbf{z}$,$\mathbf{c}$)

Check that $\mathbf{z}$ is "small"

and

$\mathbf{c} = H(\mathbf{A}\mathbf{z} - \mathbf{T}\mathbf{c} \bmod q, \mu)$

Security Reduction

Simulator | Adversary

$\xrightarrow{\mathbf{A}}$

Pick random $\mathbf{S}$

$\xrightarrow{\mathbf{A},\mathbf{AS}}$

$(z_i,c_i)=\mathrm{Sign}(\mu_i)$

$\xleftarrow{\mu_i}$

$\xrightarrow{(z_i,c_i)}$

…

$\mathbf{A(z-z')+T(c'-c)=0}$

$\xleftarrow{\mu,\ (z,c)}$

$\mathbf{A}\underbrace{(\mathbf{z-z'+Sc'-Sc})}\mathbf{=0}$

$\xleftarrow{\mu,\ (z',c')}$

If this is not 0, then **SIS** is solved.
Important for adversary to not know **S**.

Security Reduction

$$\mathbf{A}\underbrace{(\mathbf{z}-\mathbf{z}'+\mathbf{S}\mathbf{c}'-\mathbf{S}\mathbf{c})}_{\text{Solution to } \mathbf{SIS}}=\mathbf{0}$$

<u>We Want:</u>

1. Signature ($\mathbf{z},\mathbf{c}$) to be independent of $\mathbf{S}$ so that $\mathbf{z}-\mathbf{z}'+\mathbf{S}\mathbf{c}'-\mathbf{S}\mathbf{c}$ is not 0

2. $\mathbf{z}-\mathbf{z}'+\mathbf{S}\mathbf{c}'-\mathbf{S}\mathbf{c}$ to be small so that **SIS** is hard

INTERLUDE: BASING SCHEMES ON LWE INSTEAD OF SIS [L '12]

Security Reduction Requirements

Secret Key: $\mathbf{S}$

Given the public key, the secret key is not unique

Public Key: $\mathbf{A}, \mathbf{T}=\mathbf{A}\mathbf{S} \bmod q$

<u>Sign</u>(μ)

Pick a random $\mathbf{y}$

Compute $\mathbf{c}=H(\mathbf{A}\mathbf{y} \bmod q,\mu)$

$\mathbf{z}=\mathbf{S}\mathbf{c}+\mathbf{y}$

Output($\mathbf{z}$,$\mathbf{c}$) (or reject)

Signature is independent of the secret key

<u>Verify</u>($\mathbf{z}$,$\mathbf{c}$)

Check that $\mathbf{z}$ is "small"

and

$\mathbf{c} = H(\mathbf{A}\mathbf{z} - \mathbf{T}\mathbf{c} \bmod q, \mu)$

Security Reduction Requirements

Secret Key: $\mathbf{S}$

~~*Given the public key, the secret key is not unique*~~

Public Key: $\mathbf{A}$, $\mathbf{T}=\mathbf{A}\mathbf{S}$ mod q

Given the public key, it's *computationally indistinguishable* whether the secret key is unique

Sign(μ)

Pick a random $\mathbf{y}$

Compute $\mathbf{c}=H(\mathbf{A}\mathbf{y} \bmod q, \mu)$

$\mathbf{z}=\mathbf{S}\mathbf{c}+\mathbf{y}$

Output($\mathbf{z}$,$\mathbf{c}$) (or reject)

Signature is independent of the secret key

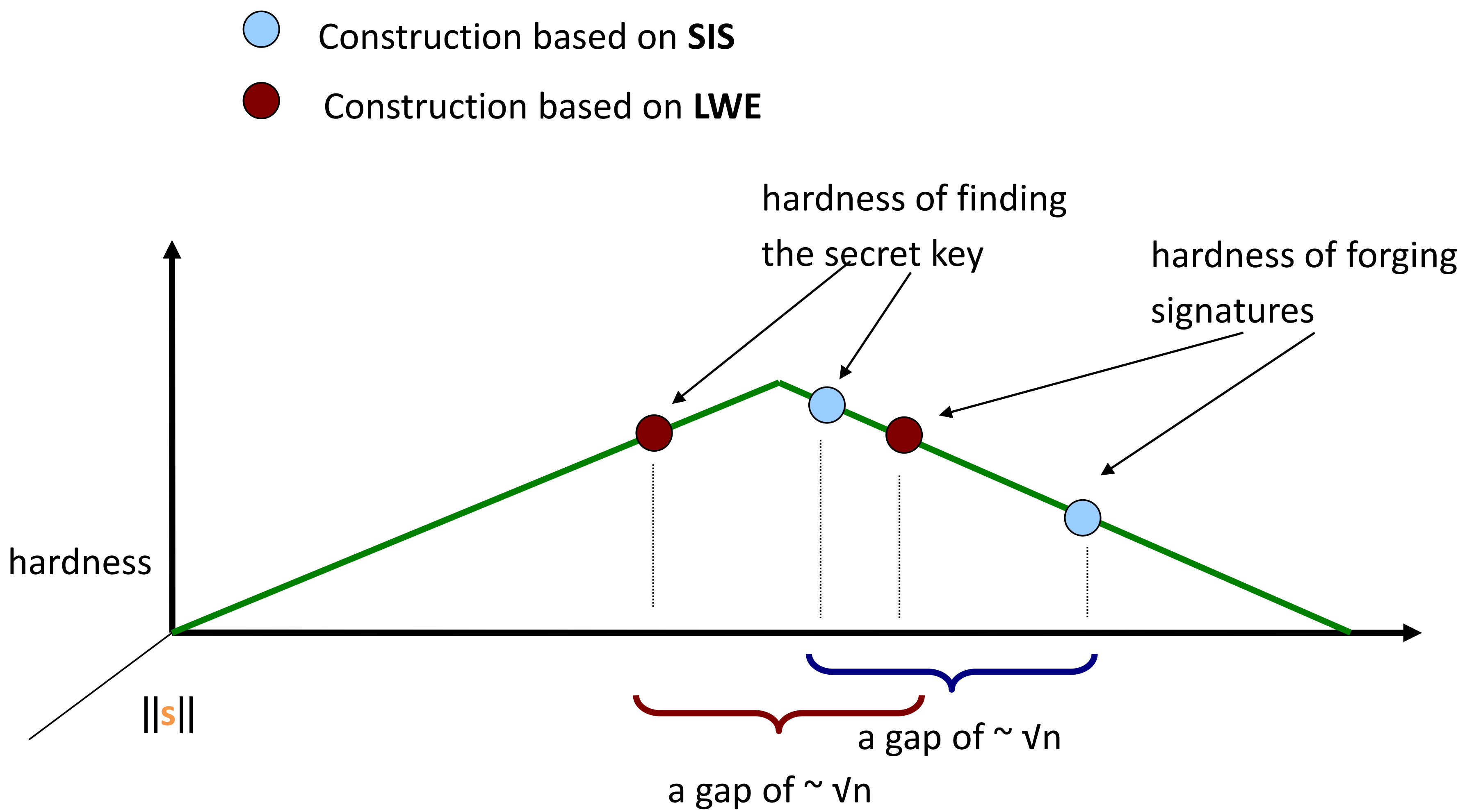
Verify($\mathbf{z}$,$\mathbf{c}$)

Check that $\mathbf{z}$ is "small"

and

$\mathbf{c} = H(\mathbf{A}\mathbf{z} - \mathbf{T}\mathbf{c} \bmod q, \mu)$

Signature Hardness



Signature Scheme

Secret Key: **S**

Public Key: **A**, **T**=**AS** mod q

<u>Sign</u>(μ)

Pick a random **y** make **y** uniformly random mod q?

Compute **c**=H(**Ay** mod q,μ)

z=**Sc**+**y**

Output(**z**,**c**) then **z** is too big and **SIS** (and forging) is easy ☹

Signature Scheme

Secret Key: **S**

Public Key: **A**, **T**=**AS** mod q

<u>Sign</u>(μ)

Pick a random **y** make **y** small?

Compute **c**=H(**Ay** mod q,μ)

z=**Sc**+**y**

Output(**z**,**c**) then **z** will not be independent of **S** ☹

Rejection Sampling

Secret Key: **S**

Public Key: **A**, **T**=**A**S mod q

<u>Sign</u>(μ)

Pick a random **y** make **y** small

Compute **c**=H(**A**y mod q,μ)

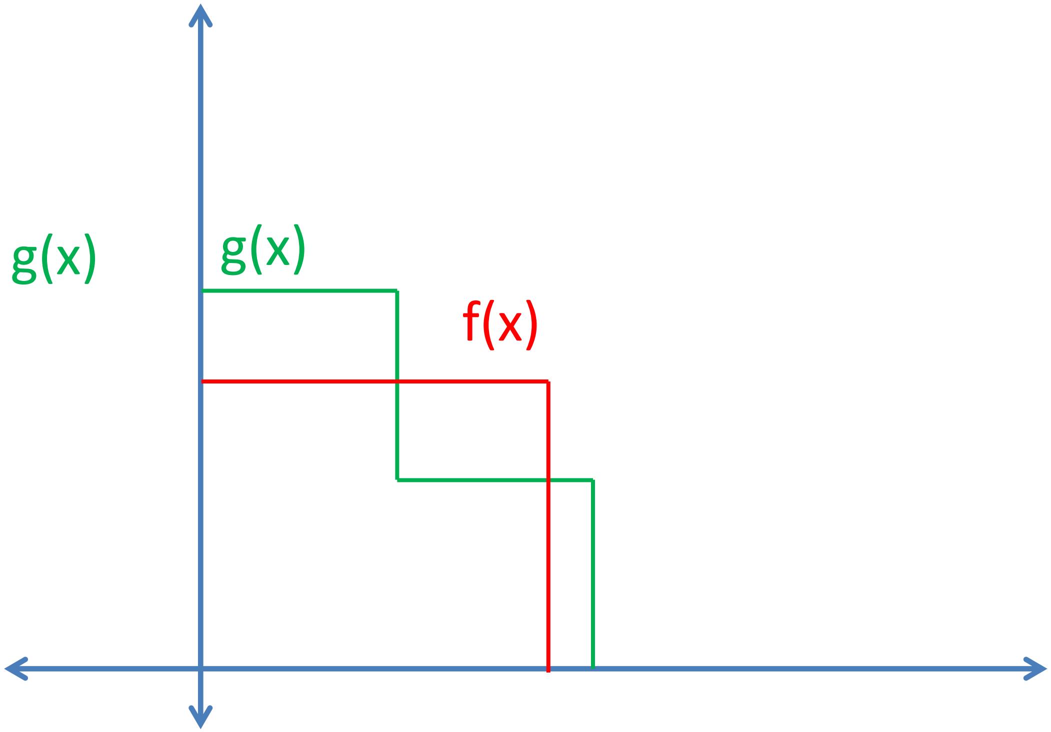
z=**Sc**+**y**

Output(**z**,**c**) *if **z** meets certain criteria, else repeat*

Rejection Sampling

Have access to samples from g(x)

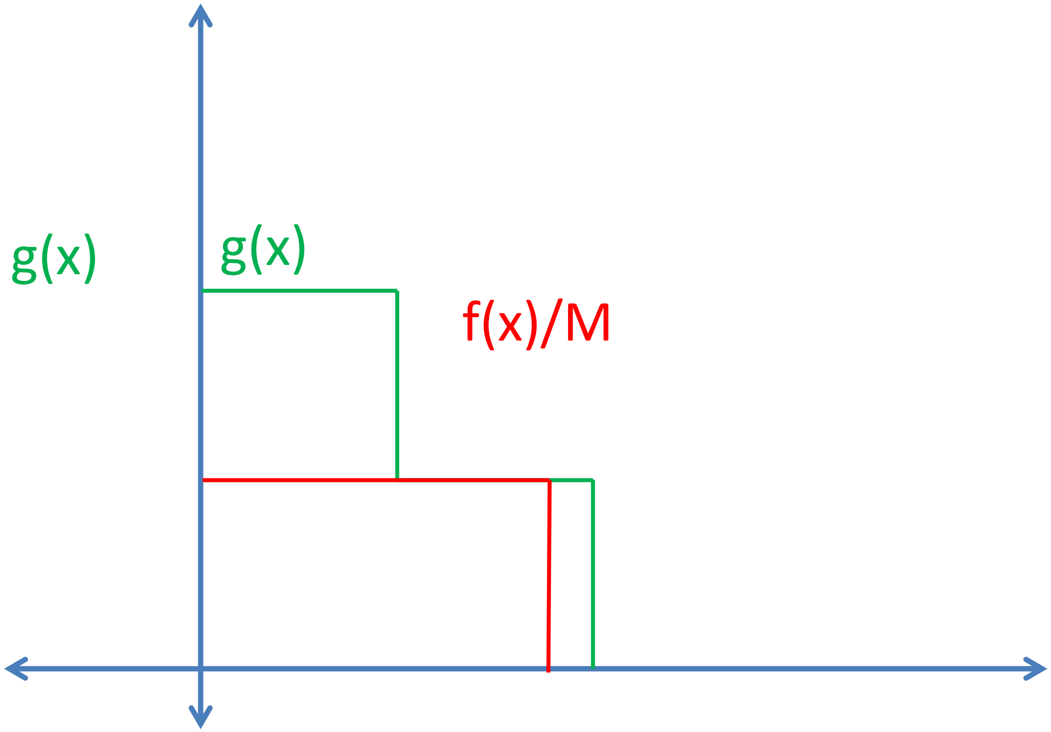
Want f(x)



Rejection Sampling

Have access to samples from g(x)

Want f(x)



Sample from g(x), accept x with probability $f(x)/Mg(x) \leq 1$

$\Pr[x] = g(x)\cdot(f(x)/Mg(x)) = f(x)/M$

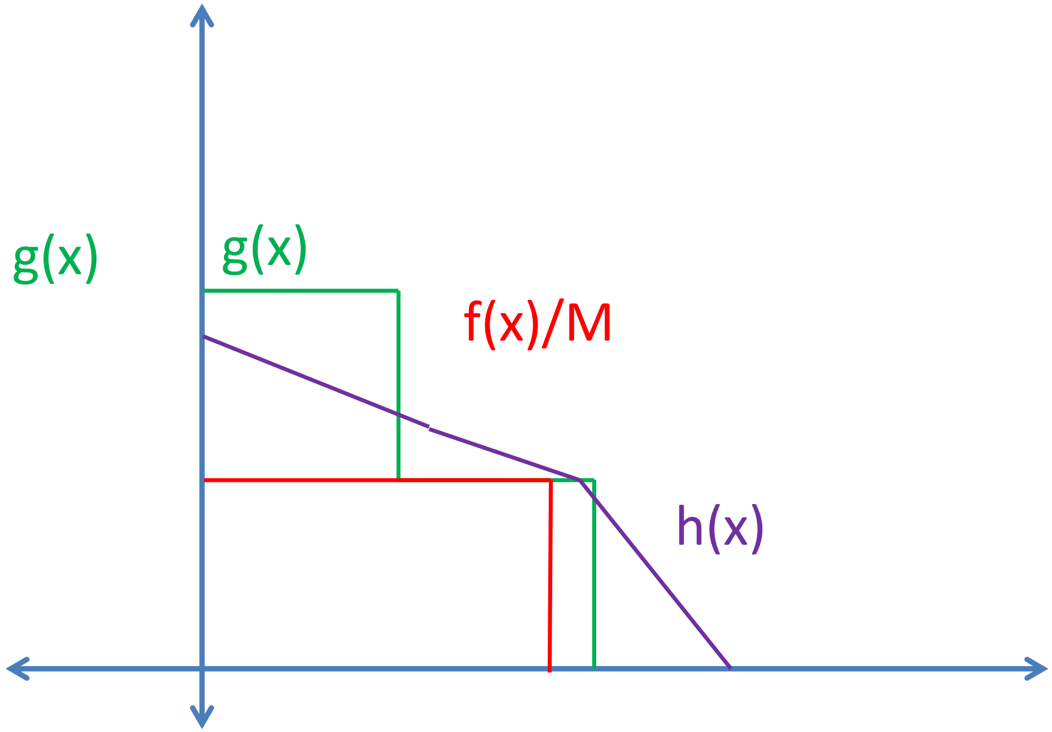
Something is output with probability 1/M

Rejection Sampling

Impossible to tell whether g(x) or h(x) was the original distribution

Have access to samples from g(x)

Want f(x)



Sample from g(x), accept x with probability $f(x)/Mg(x) \leq 1$

or … Sample from h(x), accept x with probability $f(x)/Mh(x) \leq 1$

$\Pr[x] = g(x) \cdot (f(x)/Mg(x)) = f(x)/M = h(x) \cdot (f(x)/Mh(x))$

Something is output with probability 1/M

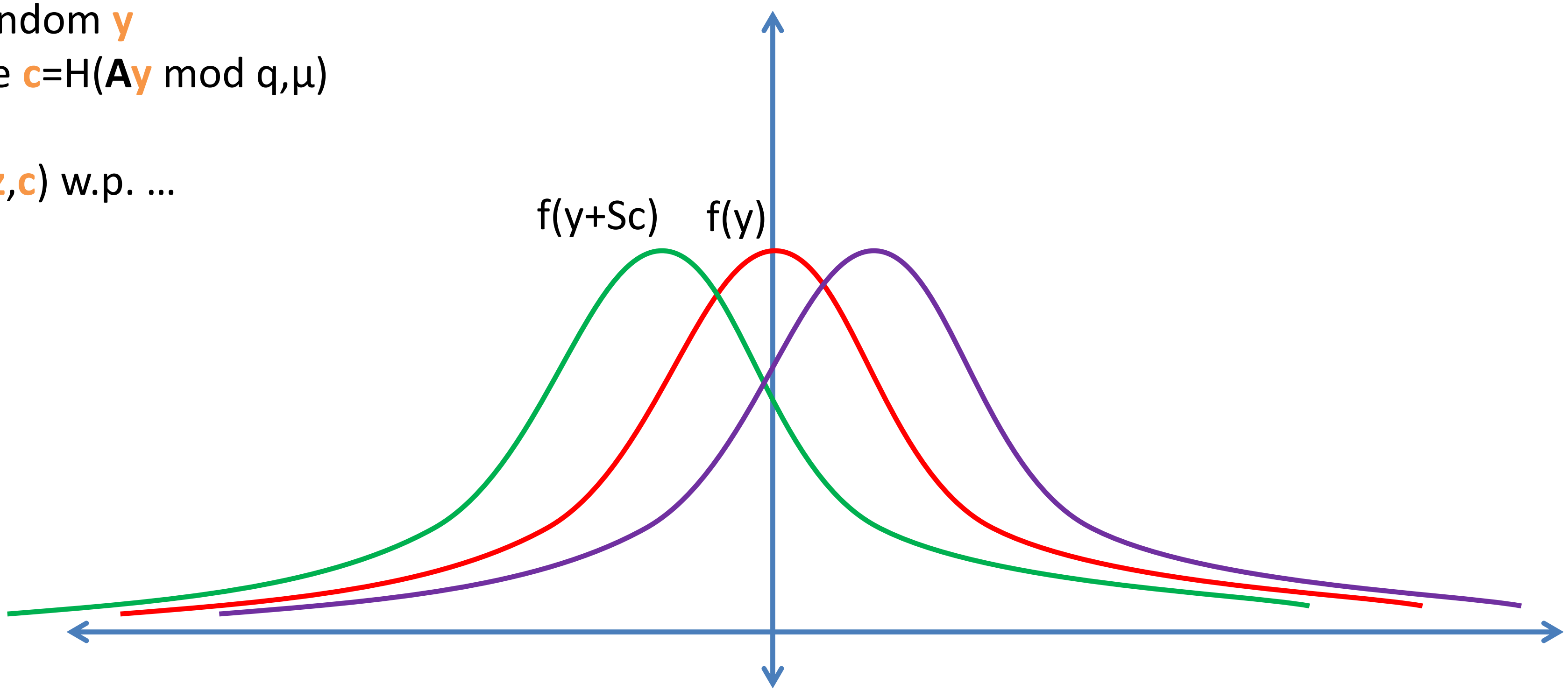
Rejection Sampling

Pick a random **y**

Compute **c**=H(**Ay** mod q,μ)

z=**Sc**+**y**

Output(**z**,**c**) w.p. …



Normal Distribution

1-dimensional Normal distribution:

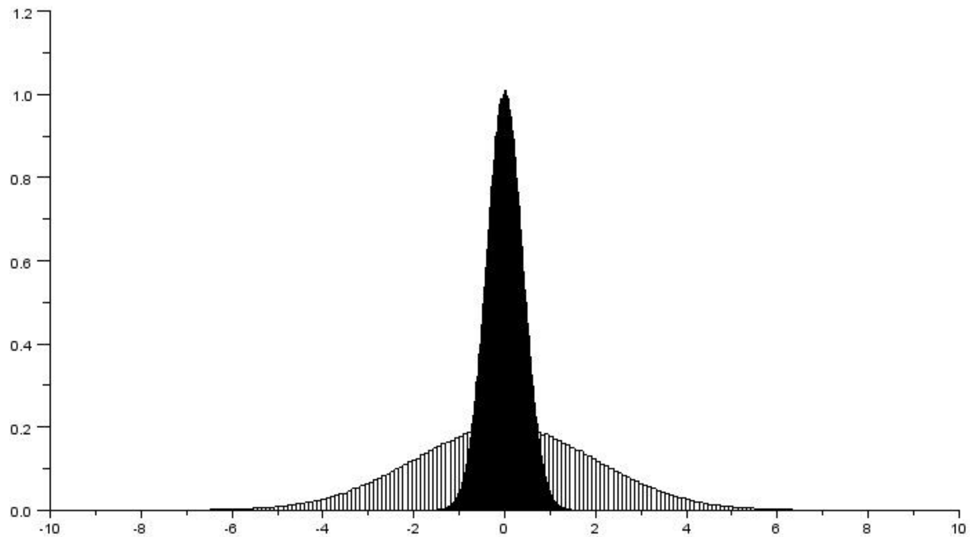
$$\rho_\sigma(x) = 1/(\sqrt{2\pi}\sigma)e^{-x^2/2\sigma^2}$$

It is:

Centered at 0

Standard deviation: σ

Examples



Shifted Normal Distribution

1-dimensional shifted Normal distribution:

$$\rho_{\sigma,v}(x) = 1/(\sqrt{2\pi}\sigma)e^{-(x-v)^2/2\sigma^2}$$

It is:

Centered at v

Standard deviation: σ

n-Dimensional Normal Distribution

n-dimensional shifted Normal distribution:

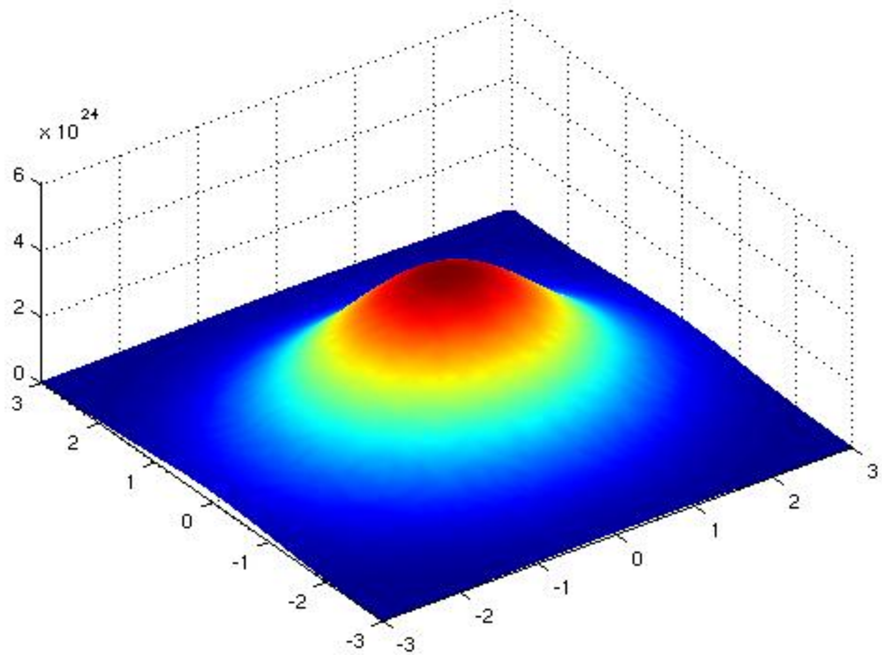
$$\rho_{\sigma,\mathbf{v}}(\mathbf{x}) = 1/(\sqrt{2\pi}\sigma)^n e^{-||\mathbf{x}-\mathbf{v}||^2/2\sigma^2}$$

It is:

Centered at **v**

Standard deviation: σ

2-Dimensional Example



n-Dimensional Normal Distribution

n-dimensional shifted Normal distribution:

$$\rho_{\sigma,\mathbf{v}}(\mathbf{x}) = 1/(\sqrt{2\pi}\sigma)^n e^{-||\mathbf{x}-\mathbf{v}||^2/2\sigma^2}$$

It is:

Centered at **v**

Standard deviation: σ

Discrete Normal: for **x** in $\mathbf{Z}^n$,

$$D_{\sigma,\mathbf{v}}\ (\mathbf{x}) = \rho_{\sigma,\mathbf{v}}(\mathbf{x}) \ /\ \rho_{\sigma,\mathbf{v}}(\mathbf{Z}^n)$$

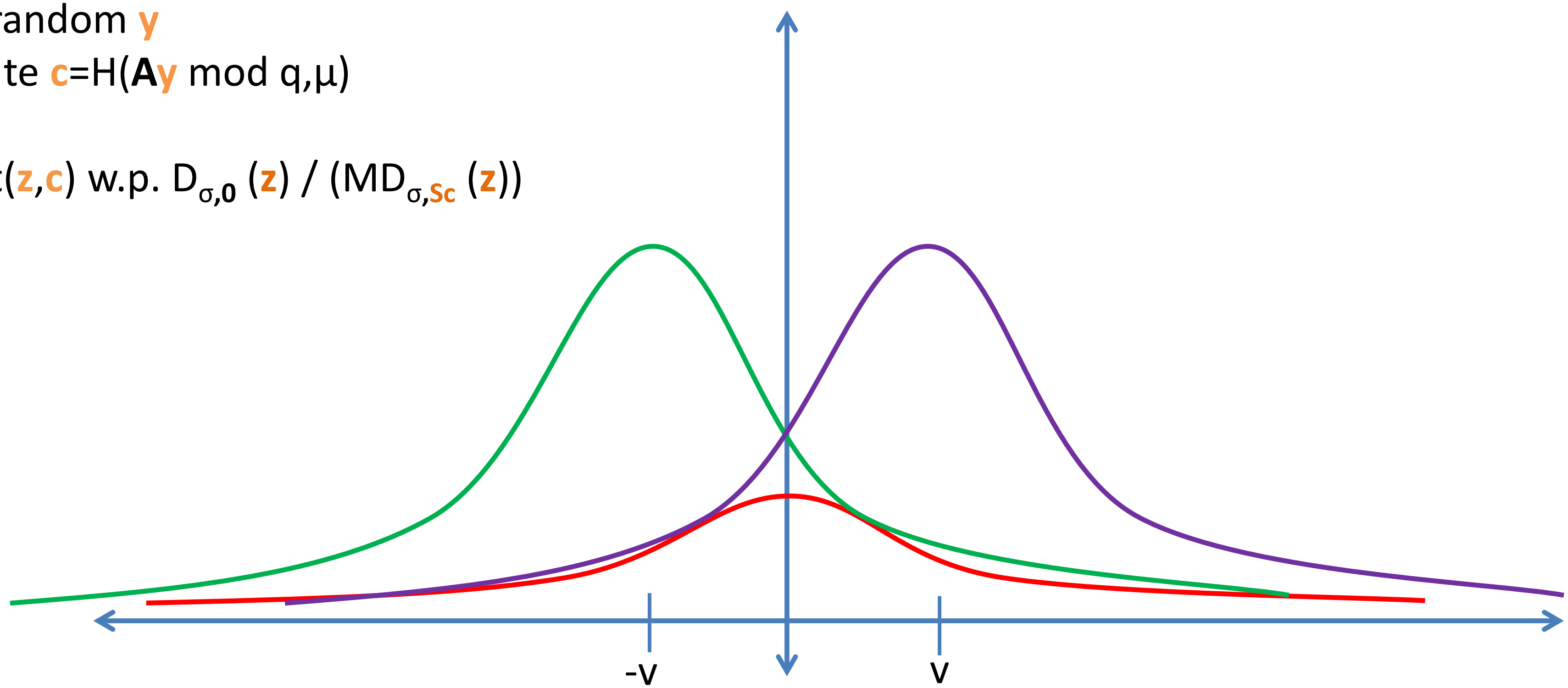
Rejection Sampling

Pick a random y

Compute $c=H(\mathbf{A}y \bmod q,\mu)$

$z=Sc+y$

Output(z,c) w.p. $D_{\sigma,\mathbf{0}}(z) / (MD_{\sigma,Sc}(z))$



v=max ||Sc||

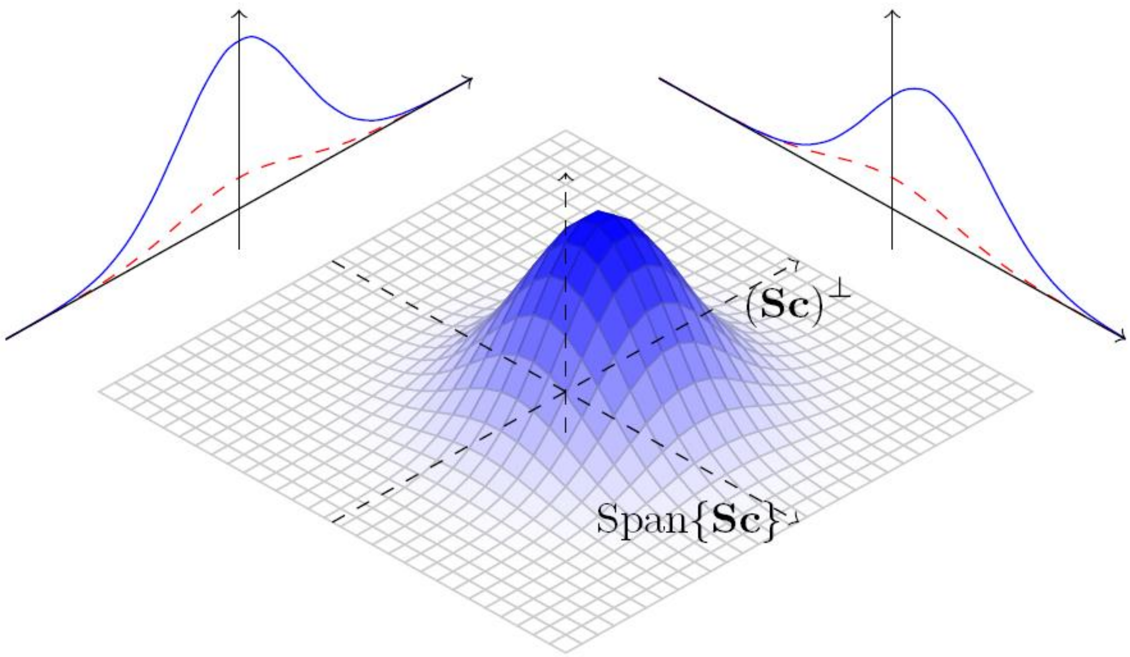
for σ = 12v,

$D_{\sigma,\mathbf{0}}(z) / (MD_{\sigma,Sc}(z)) \approx e/M$

Improving the Rejection Sampling

Pick a random **y**
Compute **c**=H(**Ay** mod q,μ)
z=**Sc**+**y**
Output(**z**,**c**) w.p. $D_{\sigma,\mathbf{0}}(\mathbf{z}) / (M D_{\sigma,\mathbf{Sc}}(\mathbf{z}))$

Rejection Sampling from [Lyu12]

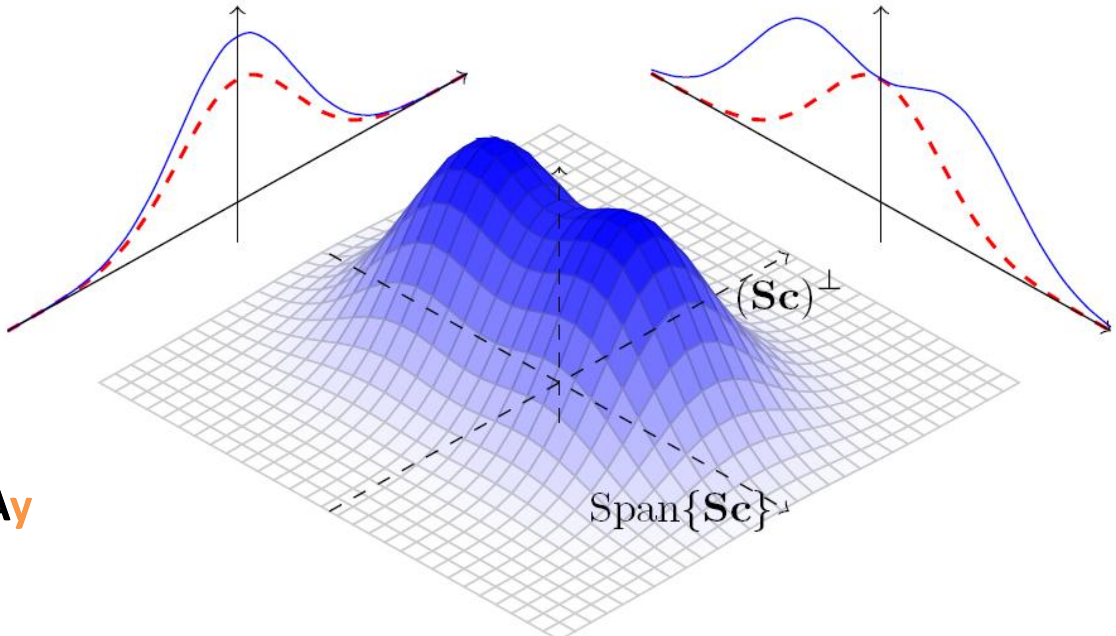


Bimodal Gaussians [DDLL '13]

Pick a random **y**
Compute **c**=H(**Ay** mod q,μ)
Pick a random b in {-1,1}
z=b**Sc**+**y**
Output(**z**,**c**) w.p. $D_{\sigma,\mathbf{0}}(\mathbf{z}) / M(½D_{\sigma,\mathbf{Sc}}(\mathbf{z}) + ½D_{\sigma,-\mathbf{Sc}}(\mathbf{z}))$

for $\sigma = \max ||\mathbf{Sc}|| / \sqrt{2}$

$$D_{\sigma,\mathbf{0}}(\mathbf{z}) / M(½D_{\sigma,\mathbf{Sc}}(\mathbf{z}) + ½D_{\sigma,-\mathbf{Sc}}(\mathbf{z})) \approx e / M$$



<u>Verify</u>(**z**,**c**)

Check that **z** is "small"

and

c = H(**Az** – **Tc** mod q, μ)

$\mathbf{Az} - \mathbf{Tc} = \mathbf{A}(b\mathbf{Sc}+\mathbf{y}) - \mathbf{Tc} = \mathbf{bTc - Tc + Ay}$

Want: $\mathbf{Tc} = -\mathbf{Tc}$

Bimodal Signature Scheme

Secret Key: $\mathbf{S}$

Public Key: $\mathbf{A}$ s.t. $q\mathbf{I}=\mathbf{AS} \bmod 2q$

<u>Sign</u>(μ)

Pick a random $\mathbf{y}$

Compute $\mathbf{c}=H(\mathbf{Ay} \bmod 2q,\mu)$

Choose random b in {-1,1}

$\mathbf{z}=b\mathbf{Sc}+\mathbf{y}$

Output($\mathbf{z}$,$\mathbf{c}$) w.p. …

<u>Verify</u>($\mathbf{z}$,$\mathbf{c}$)

Check that $\mathbf{z}$ is “small”

and

$\mathbf{c} = H(\mathbf{Az} - q\mathbf{c} \bmod 2q, \mu)$

Security Reduction

<u>Simulator</u> <u>Adversary</u>

$\xrightarrow{\mathbf{A}}$ $\xrightarrow{\mathbf{A}}$

$\xleftarrow{\mu_i}$

(z_i, c_i) ~ correct distribution

Program $c = H(\mathbf{A}z_i - qc_i \bmod 2q, \mu_i)$

$\xrightarrow{(z_i, c_i)}$

…

$$\mathbf{A(z-z')+q(c'-c)=0\ (mod\ 2q)}$$

$\xleftarrow{\mu, (z, c)}$

$$\mathbf{A}(\underbrace{z-z'})=\mathbf{0 \bmod q}$$

$\xleftarrow{\mu, (z', c')}$

If z, z' are not too small,
then this is not 0.

Optimizations

- Base problem on the hardness of the NTRU problem
- Compress the signature → not all of z needs to be output if H only acts on the high order bits
- A few other small tricks

Performance of the Bimodal LattIce Signature Scheme

Implementation	Security	Signature Size	SK Size	PK Size	Sign (ms)	Sign/s	Verify (ms)	Verify/s
BLISS-0	$\leqslant$ 60 bits	3.3 kb	1.5 kb	3.3 kb	0.241	4k	0.017	59k
BLISS-I	128 bits	5.6 kb	2 kb	7 kb	0.124	8k	0.030	33k
BLISS-II	128 bits	5 kb	2 kb	7 kb	0.480	2k	0.030	33k
BLISS-III	160 bits	6 kb	3 kb	7 kb	0.203	5k	0.031	32k
BLISS-IV	192 bits	6.5 kb	3 kb	7 kb	0.375	2.5k	0.032	31k
RSA 1024	72-80 bits	1 kb	1 kb	1 kb	0.167	6k	0.004	91k
RSA 2048	103-112 bits	2 kb	2 kb	2 kb	1.180	0.8k	0.038	27k
RSA 4096	$\geqslant$ 128 bits	4 kb	4 kb	4 kb	8.660	0.1k	0.138	7.5k
ECDSA[1] 160	80 bits	0.32 kb	0.16 kb	0.16 kb	0.058	17k	0.205	5k
ECDSA 256	128 bits	0.5 kb	0.25 kb	0.25 kb	0.106	9.5k	0.384	2.5k
ECDSA 384	192 bits	0.75 kb	0.37 kb	0.37 kb	0.195	5k	0.853	1k

THANK YOU