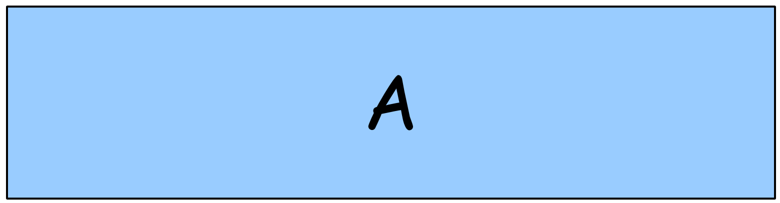
Efficient Cryptography from Generalized Compact Knapsacks

Vadim Lyubashevsky
INRIA & ENS Paris

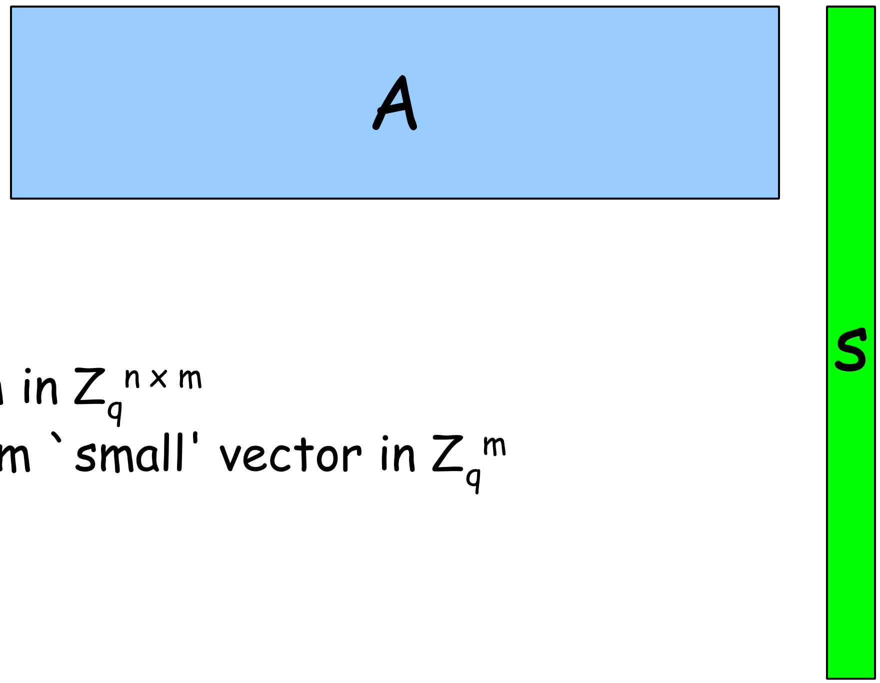
The Knapsack Problem

The Knapsack Problem



A is random in $Z_q^{n \times m}$

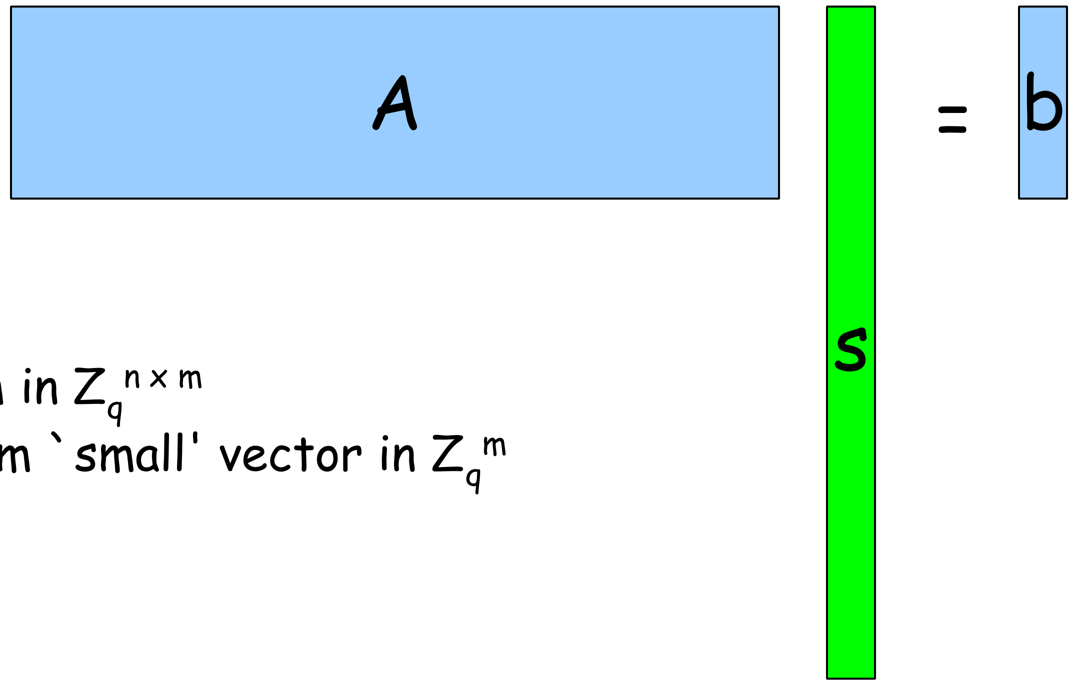
The Knapsack Problem



A is random in $Z_q^{n \times m}$

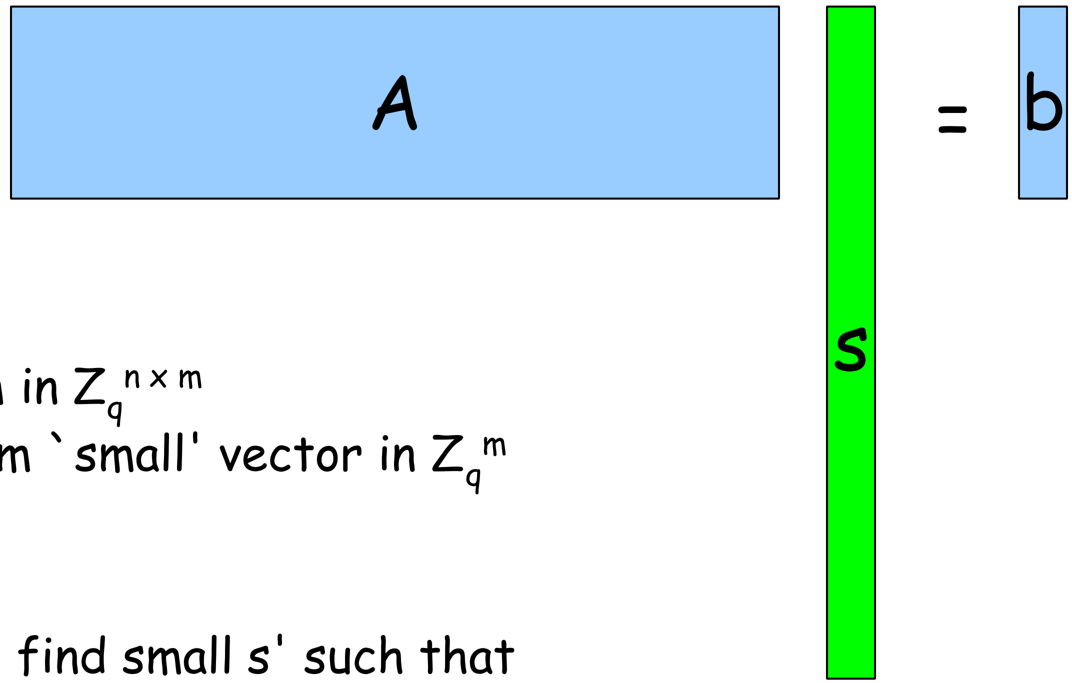
s is a random `small' vector in Z_q^m

The Knapsack Problem



A is random in $Z_q^{n \times m}$
s is a random `small' vector in Z_q^m
b=As mod q

The Knapsack Problem



A is random in $Z_q^{n \times m}$

s is a random `small' vector in Z_q^m

b=As mod q

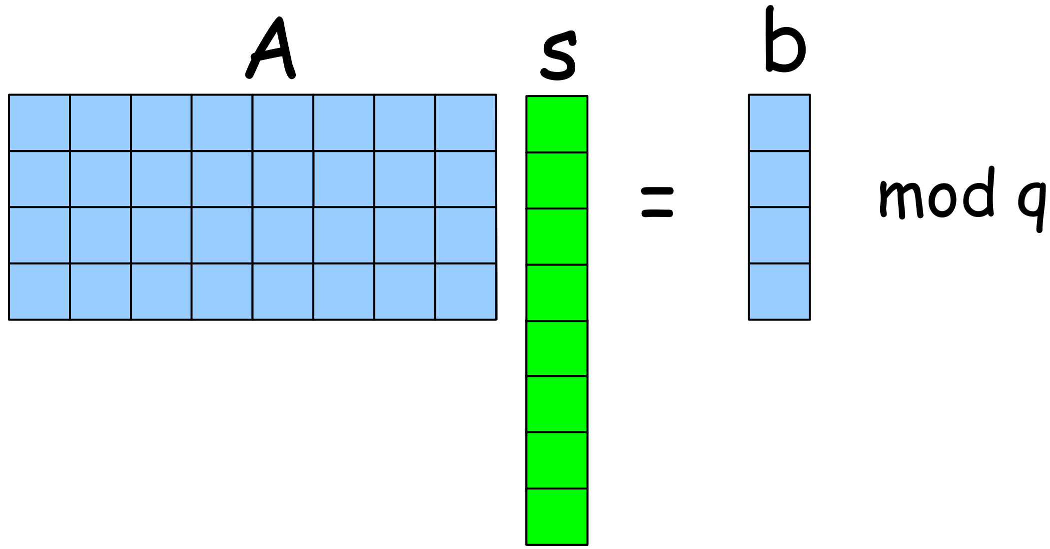
Given (A,b), find small s' such that

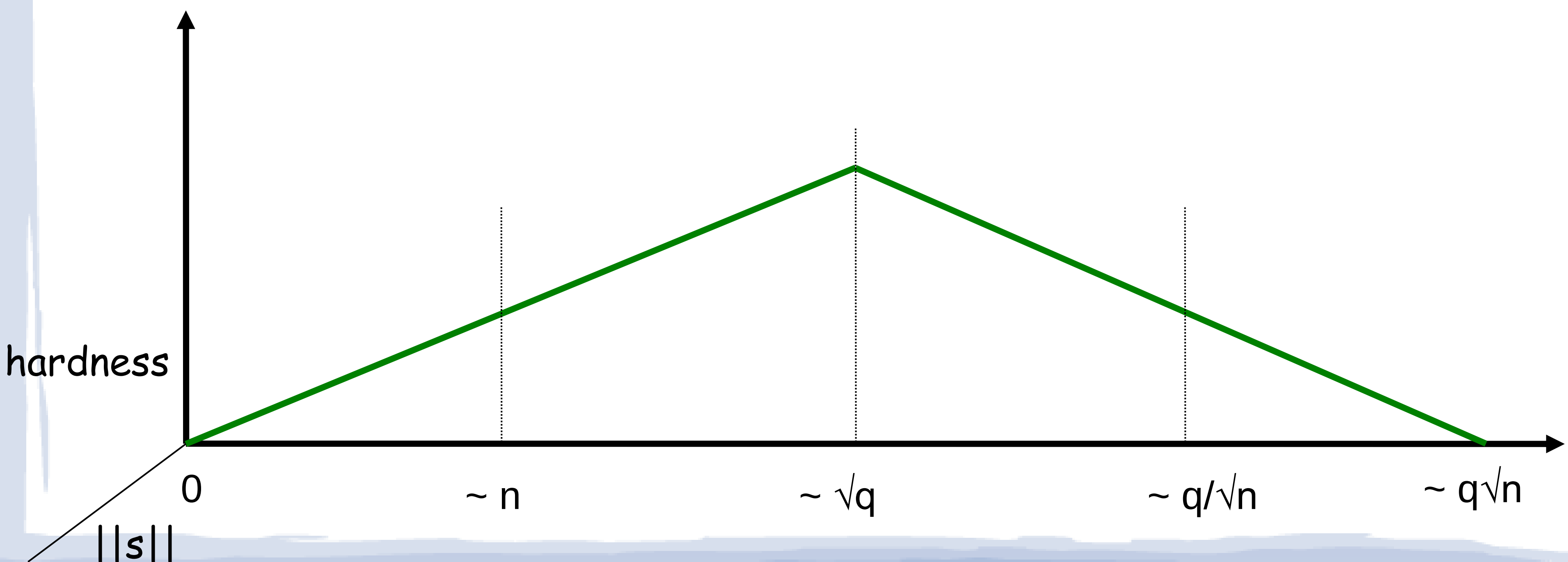
As'=b mod q

The Knapsack Problem

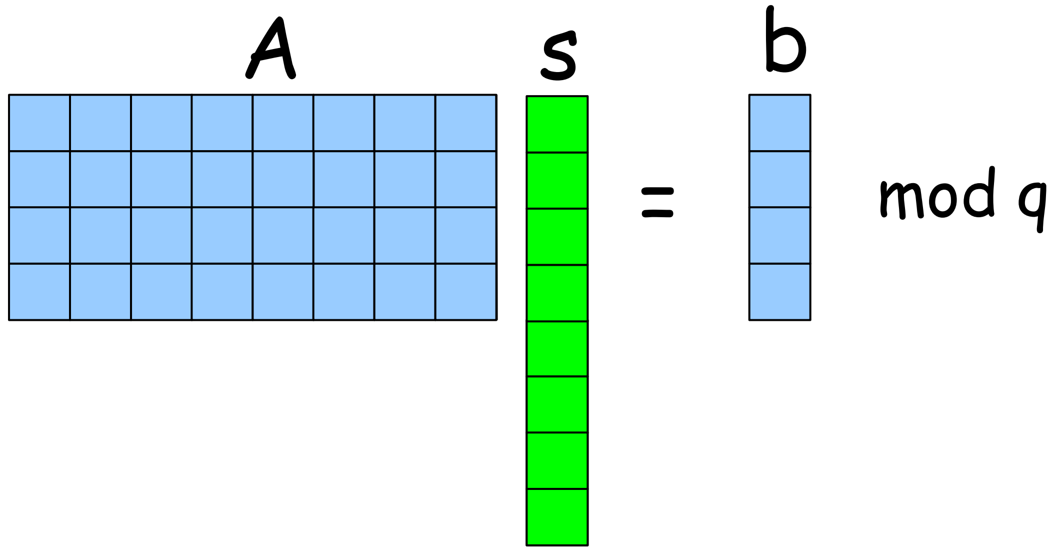
$$\overset{A}{\begin{bmatrix} 4 & 11 & 6 & 8 & 1 & 0 & 0 & 0 \\ 7 & 7 & 1 & 2 & 0 & 1 & 0 & 0 \\ 2 & 9 & 12 & 5 & 0 & 0 & 1 & 0 \\ 1 & 3 & 14 & 9 & 0 & 0 & 0 & 1 \end{bmatrix}} \overset{s}{\begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}} = \overset{b}{\begin{bmatrix} 12 \\ 10 \\ 8 \\ 10 \end{bmatrix}} \quad \text{mod } 17$$

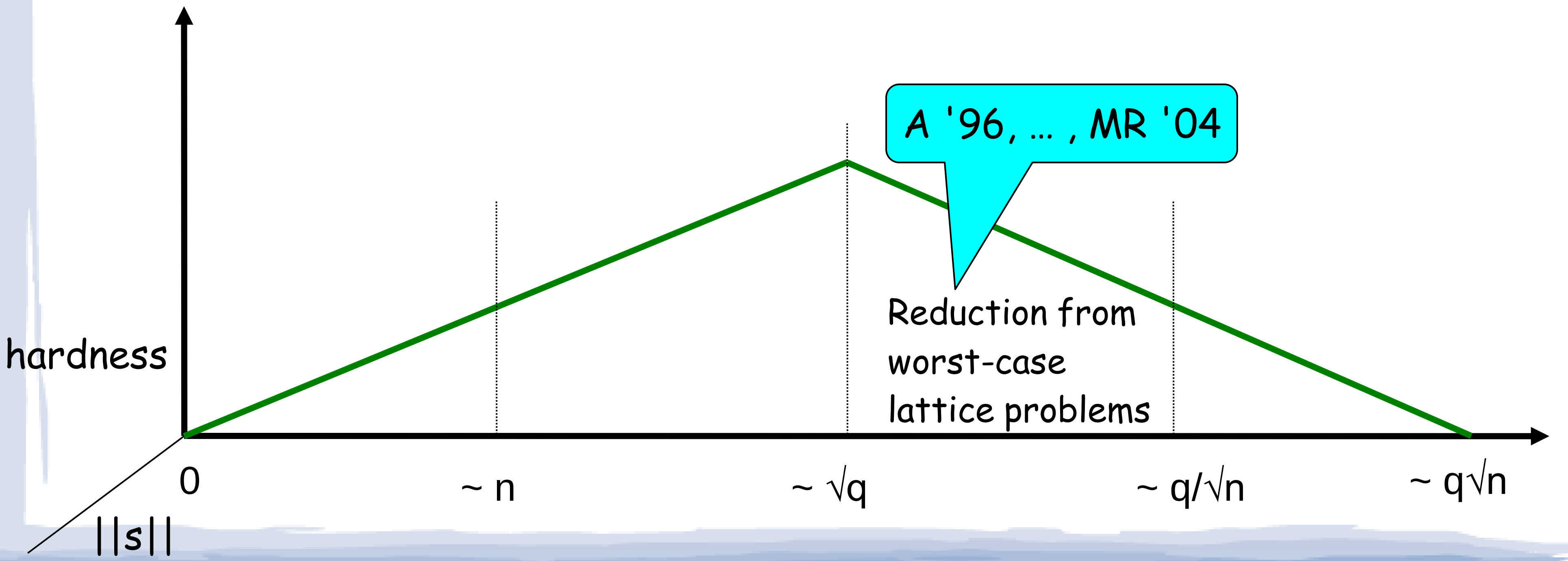
Hardness of the Knapsack Problem



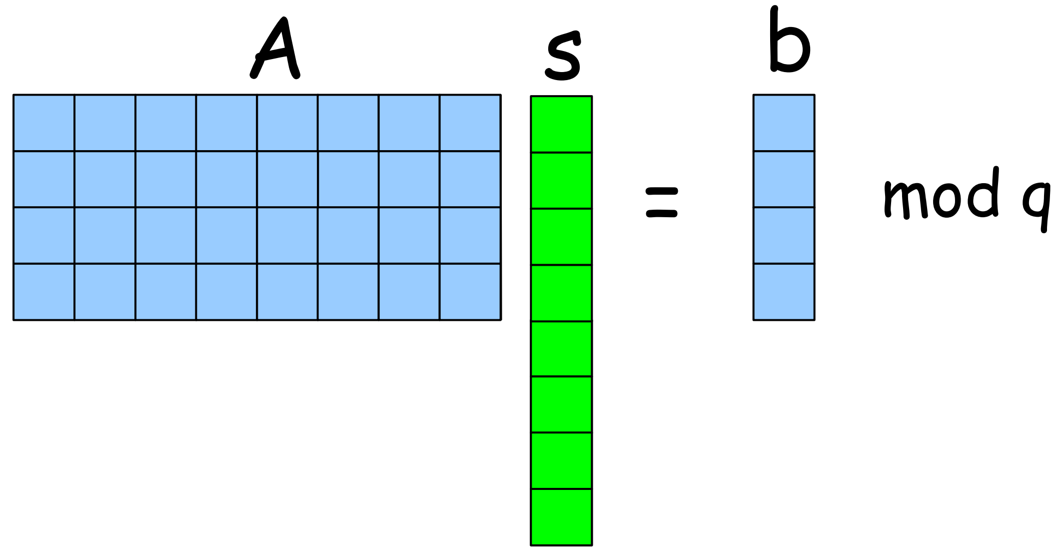


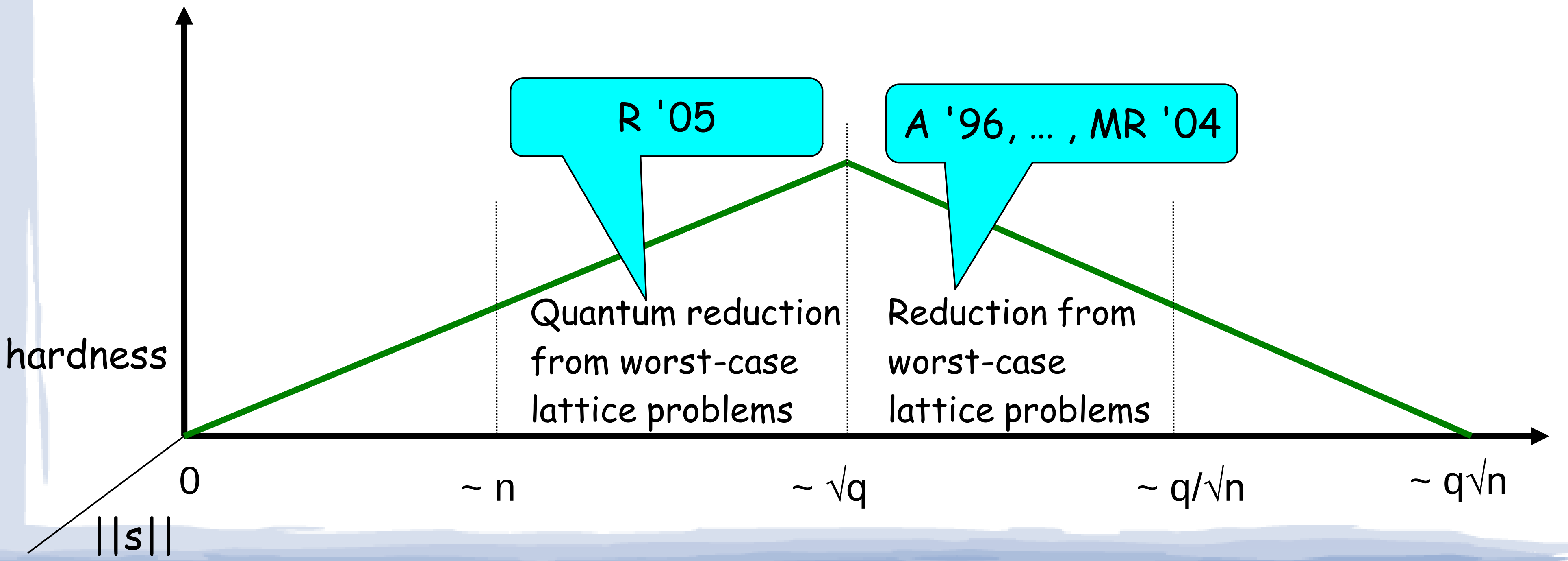
Hardness of the Knapsack Problem



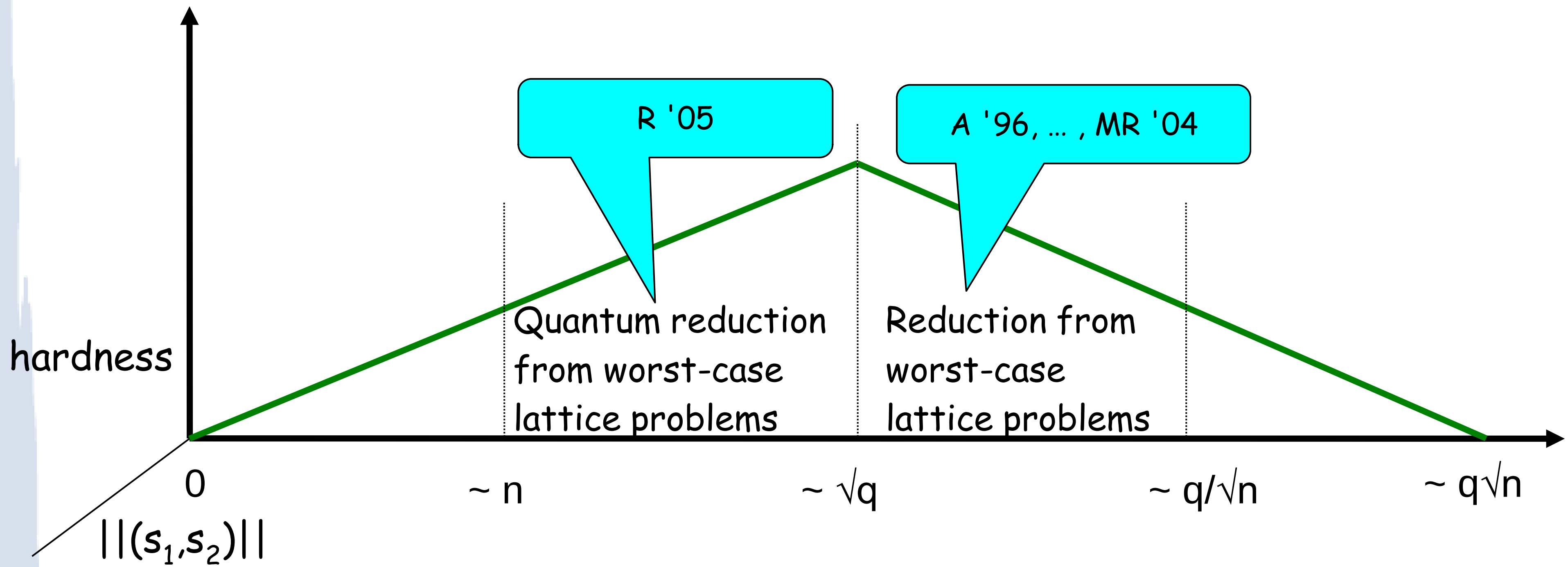


Hardness of the Knapsack Problem

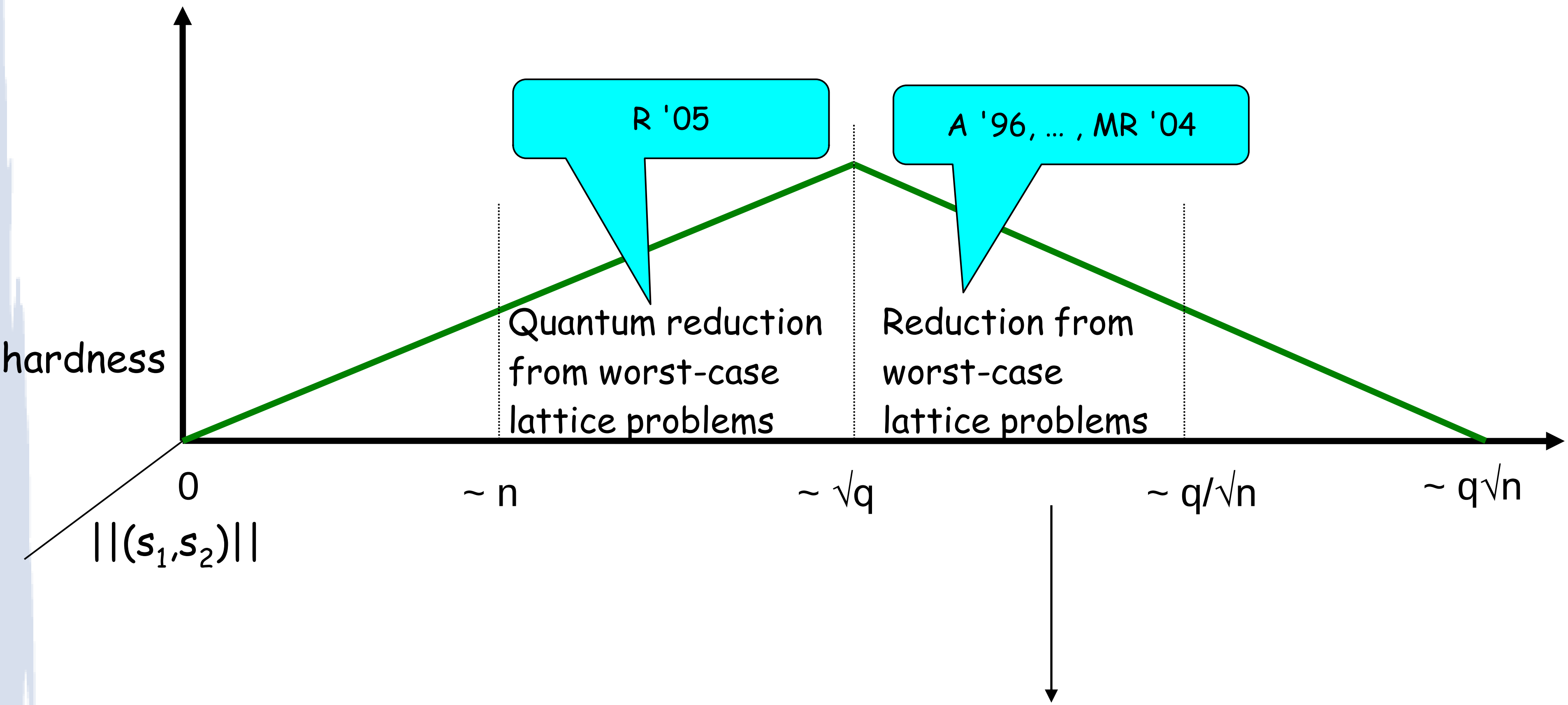




Cryptographic Primitives



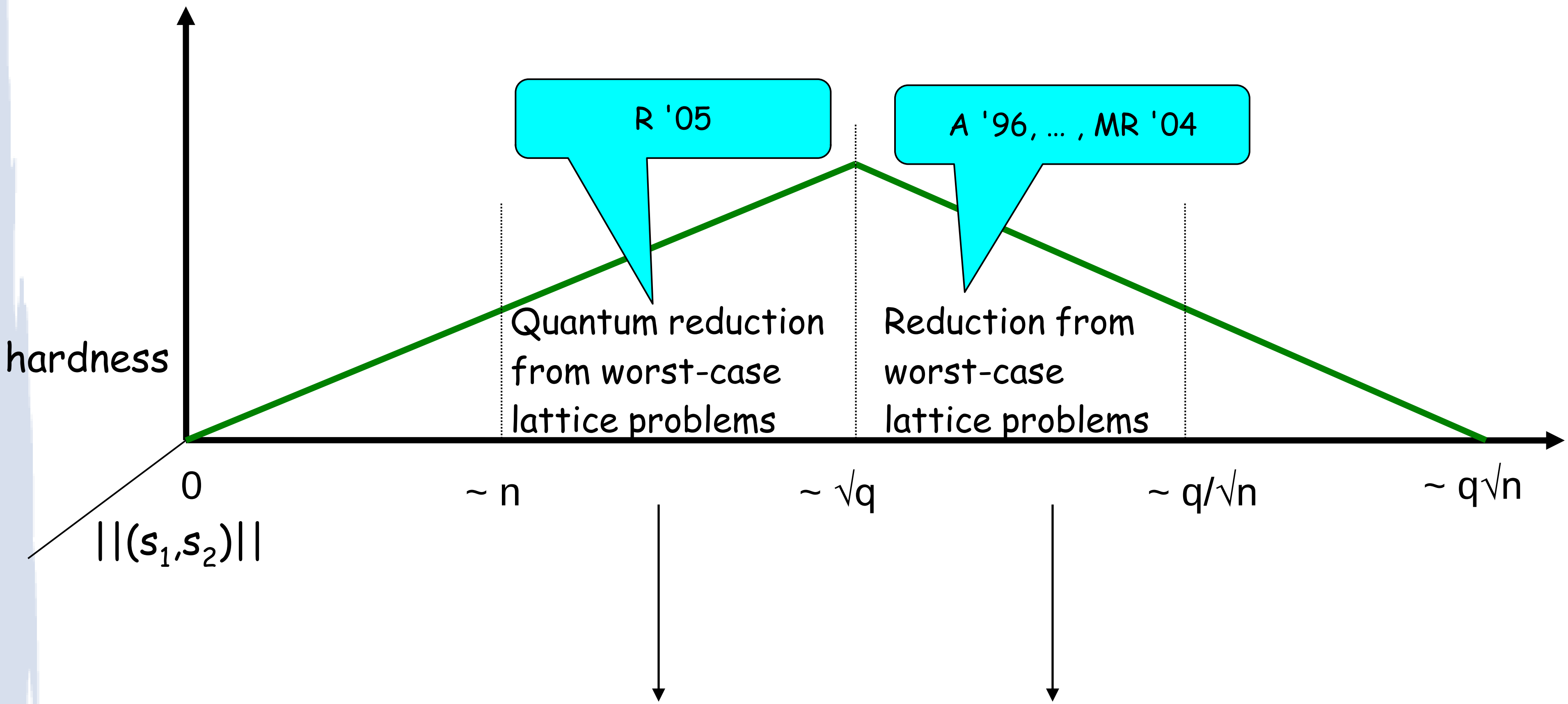
Cryptographic Primitives



- One-way functions
- Collision resistant hash functions
- Identification schemes
- Digital signatures

(minicrypt)

Cryptographic Primitives



- Public-Key Encryption
- Identity-Based Encryption

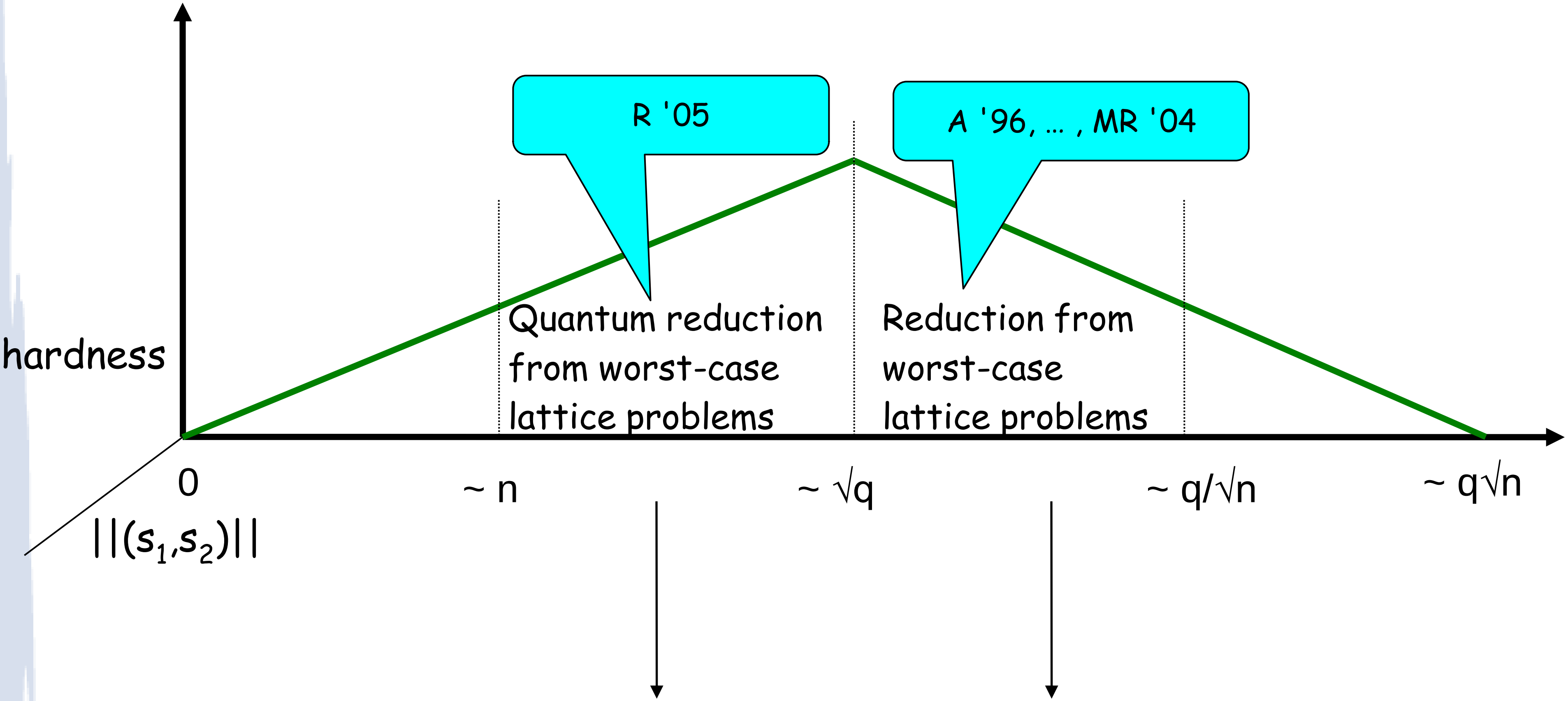
…

(cryptomania)

- One-way functions
- Collision resistant hash functions
- Identification schemes
- Digital signatures

(minicrypt)

Practical Cryptographic Primitives?



- Public-Key Encryption
- Identity-Based Encryption

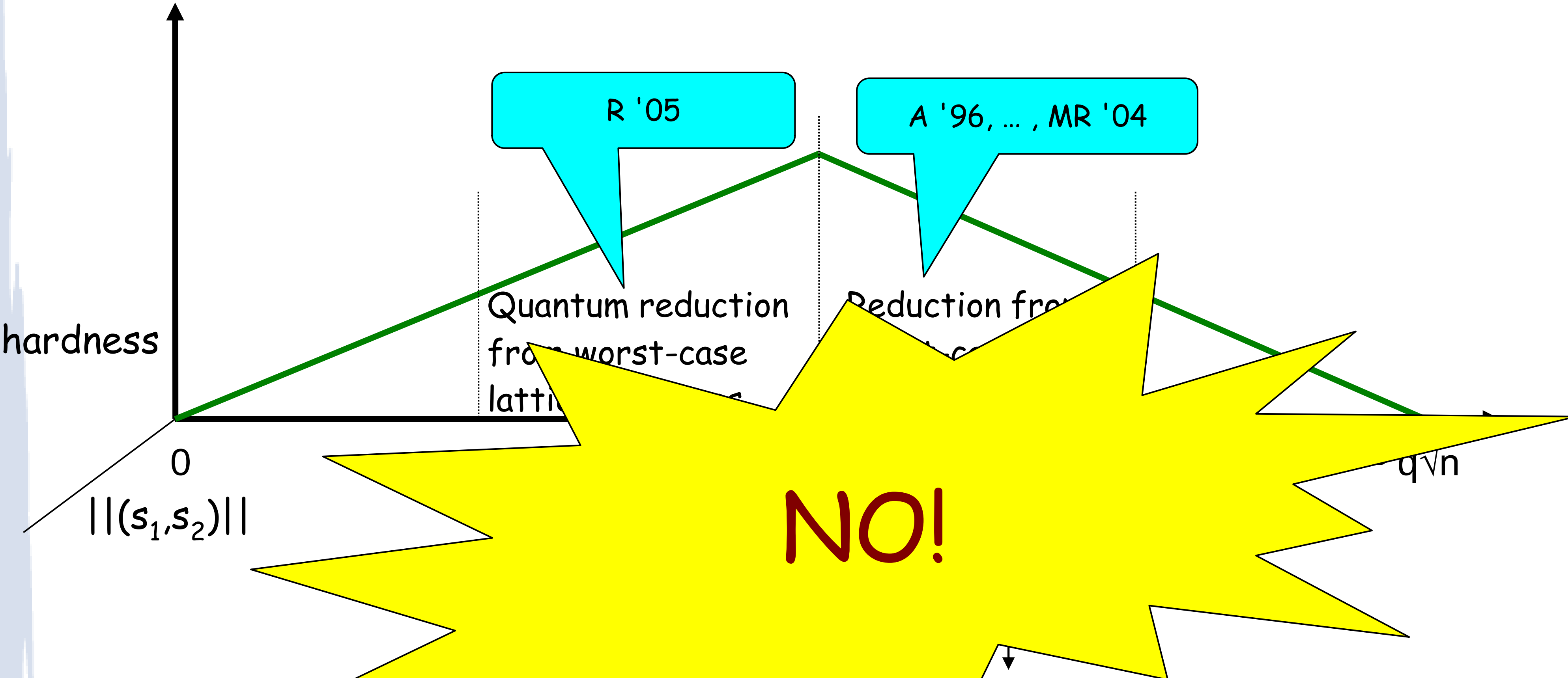
…

(cryptomania)

- One-way functions
- Collision resistant hash functions
- Identification schemes
- Digital signatures

(minicrypt)

Practical Cryptographic Primitives?



- Public-Key Er[illegible]on
- Identity-Bas[illegible] Encryption

…

(cryptomania)

- One-way functions
- Collision resistant hash functions
- Identification schemes
- Digital signatures

(minicrypt)

Why Construct Crypto Primitives Based on Knapsacks?

Why Construct Crypto Primitives Based on Knapsacks?

- Substantially different from number theoretic constructions

Why Construct Crypto Primitives Based on Knapsacks?

- Substantially different from number theoretic constructions
- Seem to resist quantum attacks

Why Construct Crypto Primitives Based on Knapsacks?

- Substantially different from number theoretic constructions
- Seem to resist quantum attacks
- Possibly faster

Why Construct Crypto Primitives Based on Knapsacks?

- Substantially different from number theoretic constructions
- Seem to resist quantum attacks
- Possibly faster
- Very interesting security guarantee

Why Construct Crypto Primitives Based on Knapsacks?

- Substantially different from number theoretic constructions
- Seem to resist quantum attacks
- Possibly faster
- Very interesting security guarantee

Can we have the same properties and practicality?

The Compact Knapsack Problem

$$\overset{A}{\begin{bmatrix} 4 & -1 & -2 & -7 & 1 & 0 & 0 & 0 \\ 7 & 4 & -1 & -2 & 0 & 1 & 0 & 0 \\ 2 & 7 & 4 & -1 & 0 & 0 & 1 & 0 \\ 1 & 2 & 7 & 4 & 0 & 0 & 0 & 1 \end{bmatrix}} \overset{s}{\begin{bmatrix} \ \\ \ \\ \ \\ \ \\ \ \\ \ \\ \ \\ \ \end{bmatrix}} = \overset{b}{\begin{bmatrix} \ \\ \ \\ \ \\ \ \end{bmatrix}} \mod q$$

The Compact Knapsack Problem

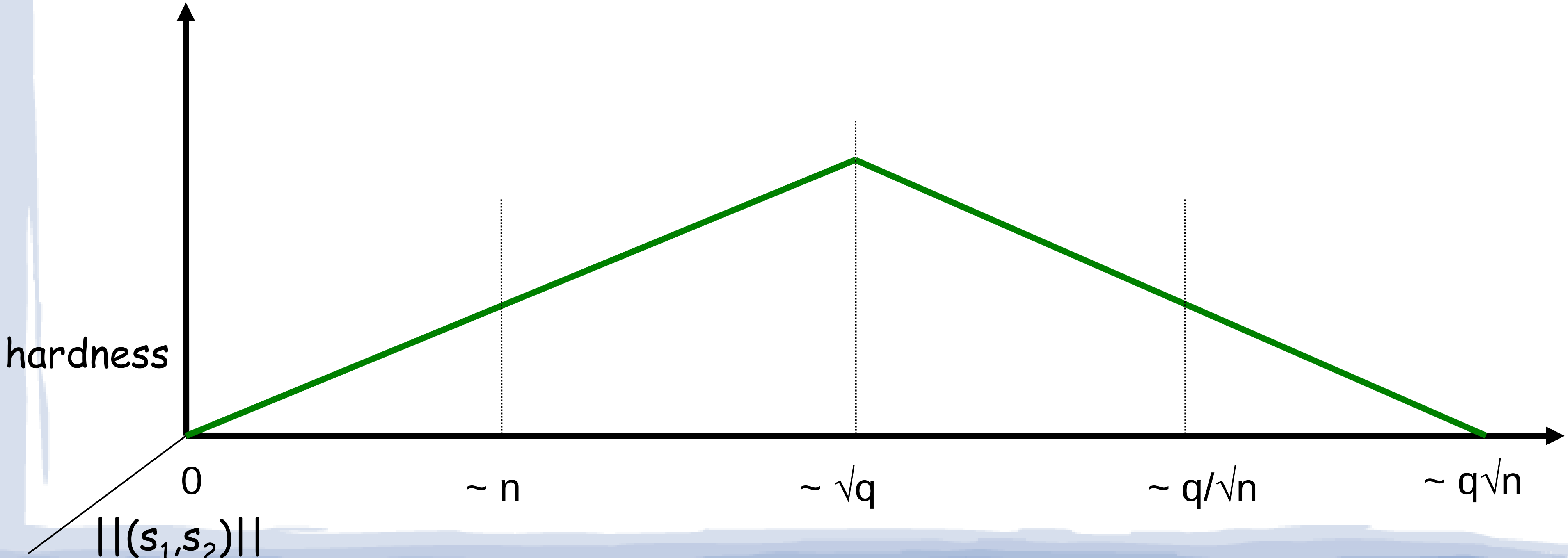
$$\overset{A}{\begin{bmatrix} 4 & -1 & -2 & -7 & 1 & 0 & 0 & 0 \\ 7 & 4 & -1 & -2 & 0 & 1 & 0 & 0 \\ 2 & 7 & 4 & -1 & 0 & 0 & 1 & 0 \\ 1 & 2 & 7 & 4 & 0 & 0 & 0 & 1 \end{bmatrix}} \overset{s}{\begin{bmatrix} \; \\ \; \\ \; \\ \; \\ \; \\ \; \\ \; \\ \; \end{bmatrix}} = \overset{b}{\begin{bmatrix} \; \\ \; \\ \; \\ \; \end{bmatrix}} \mod q$$

Equivalent to polynomial multiplication in the ring $R = Z_q[x]/(x^n+1)$

$$as_1 + s_2 = b$$

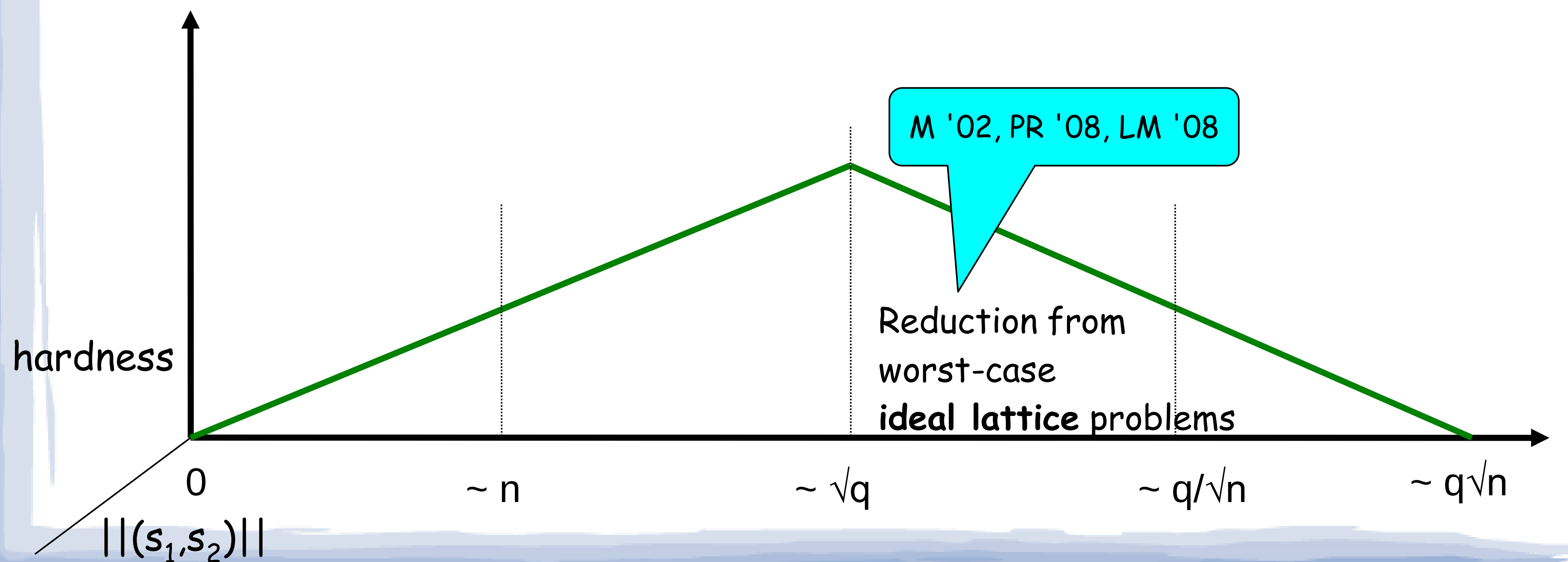
Hardness of the Compact Knapsack Problem

$$as_1 + s_2 = b \mod q$$



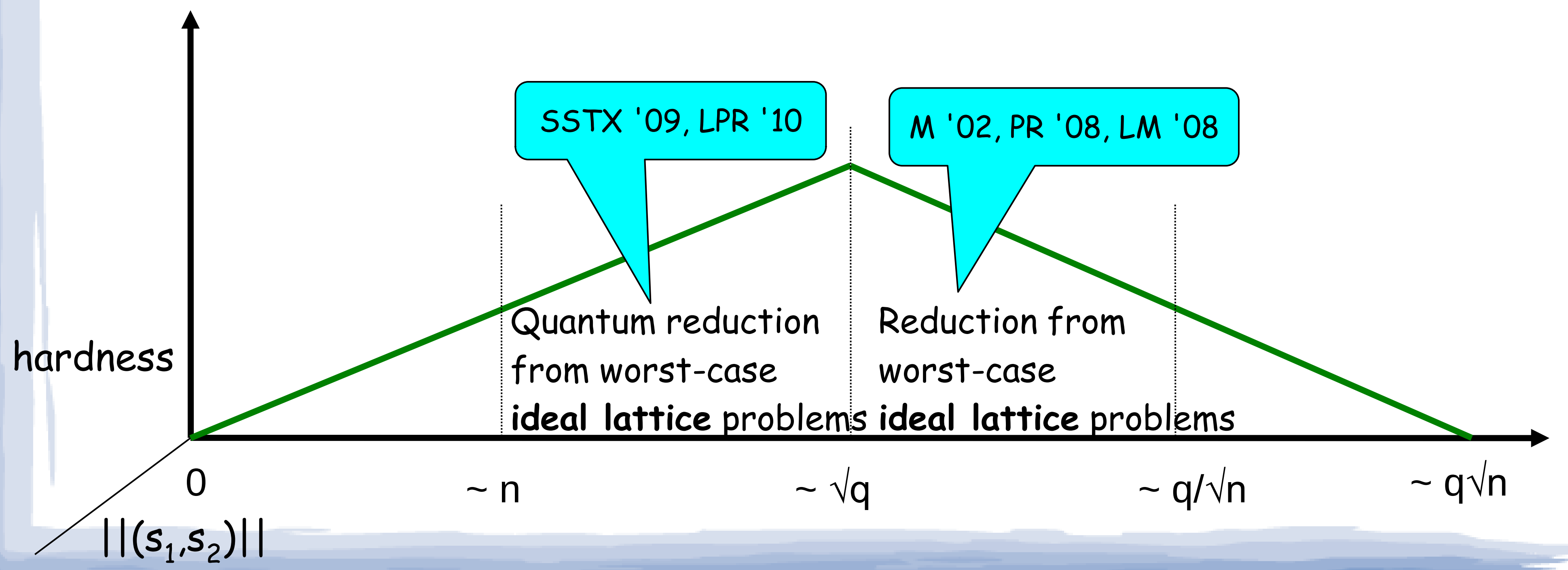
Hardness of the Compact Knapsack Problem

$$as_1 + s_2 = b \mod q$$

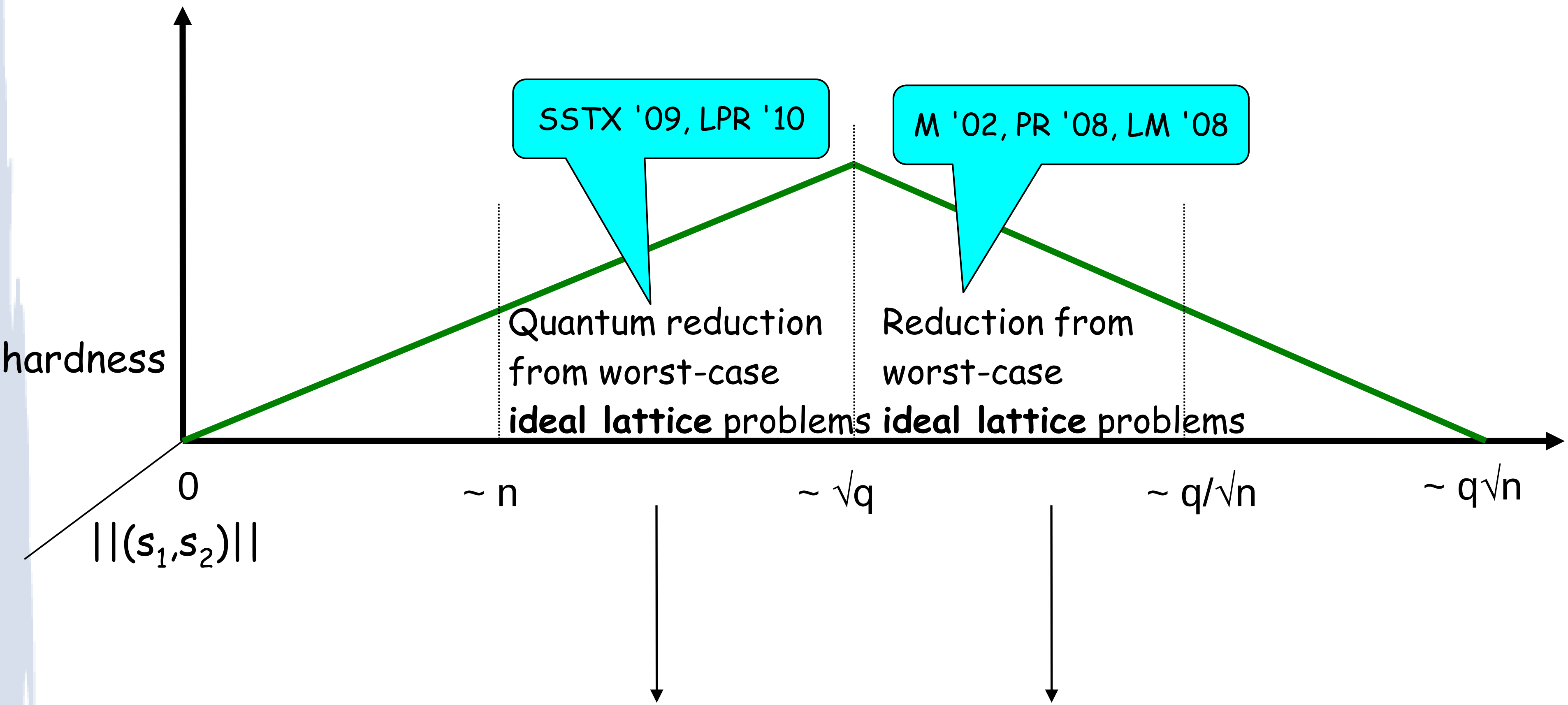


Hardness of the Compact Knapsack Problem

$$as_1 + s_2 = b \mod q$$



Cryptographic Primitives



- Public-Key Encryption
- Identity-Based Encryption
- Homomorphic Encryption

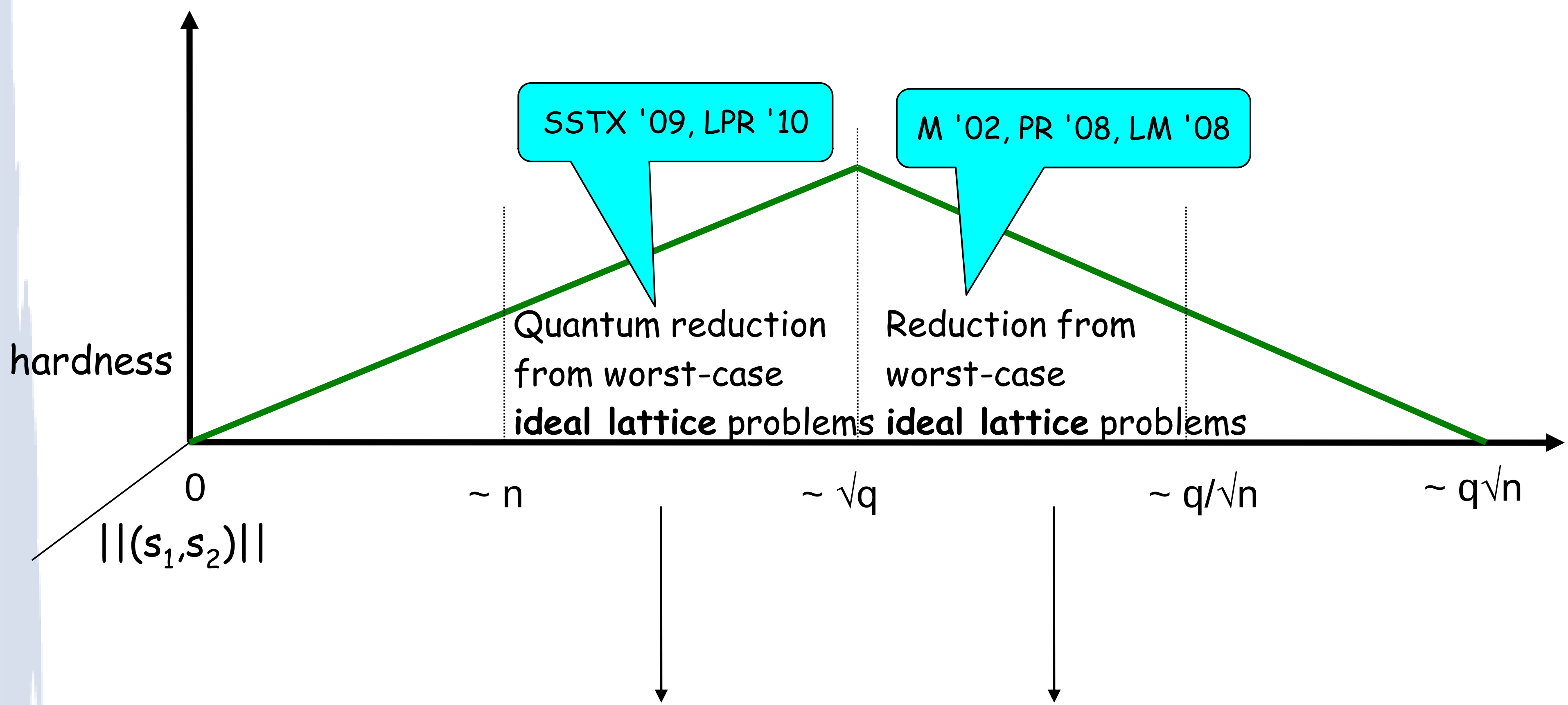
…

(cryptomania)

- One-way functions
- Collision resistant hash functions
- Identification schemes
- Digital signatures

(minicrypt)

Practical Cryptographic Primitives?



- Public-Key Encryption
- Identity-Based Encryption
- Homomorphic Encryption

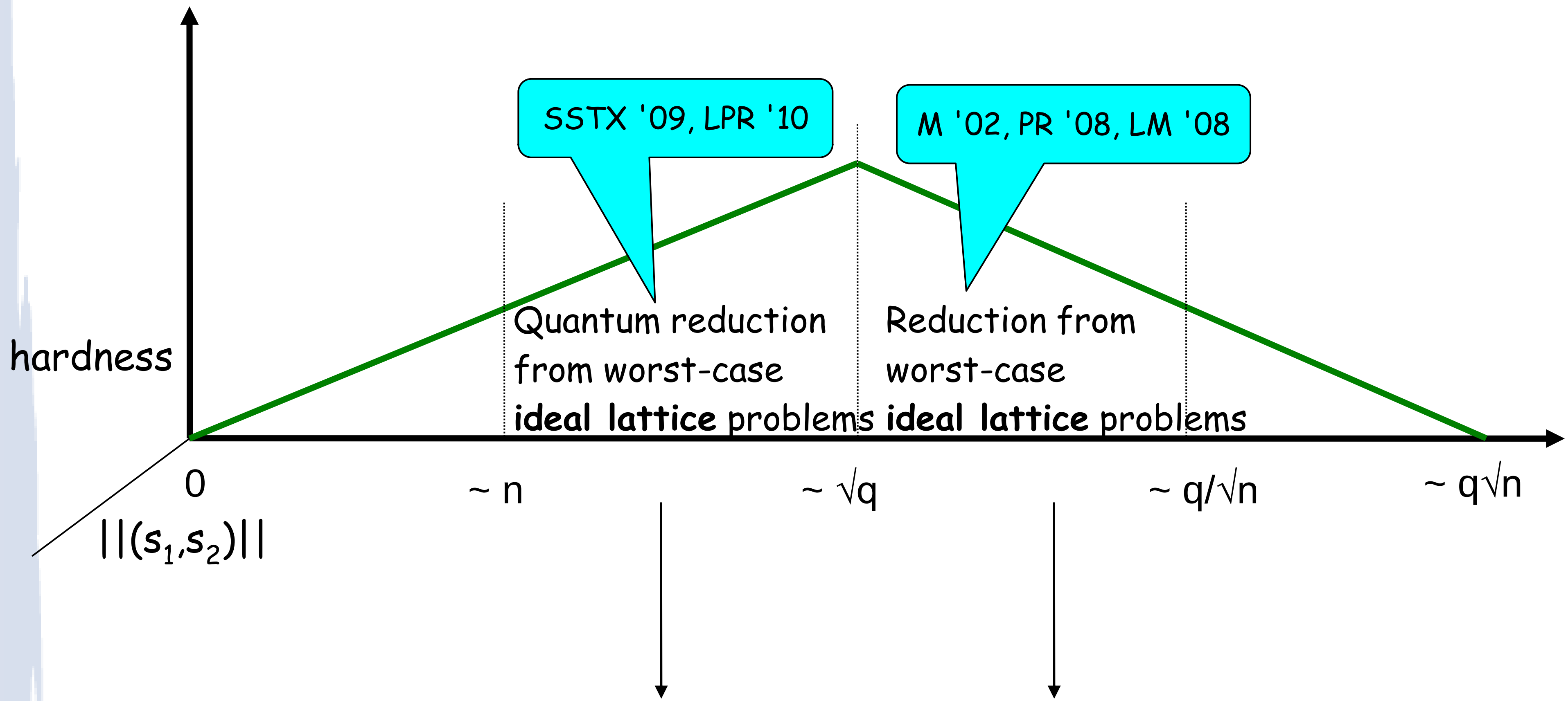
…

(cryptomania)

- One-way functions
- Collision resistant hash functions
- Identification schemes
- Digital signatures

(minicrypt)

Practical Cryptographic Primitives?



- Public-Key Encryption
- Identity-Based Encryption
- Homomorphic Encryption

…

(cryptomania)

- One-way functions
- Collision resistant hash functions
- Identification schemes
- Digital signatures

(minicrypt)

Digital Signatures

Digital Signatures

- Arguably the most important application of public key cryptography

Digital Signatures

- Arguably the most important application of public key cryptography
- Signature lengths for ~ 80 bits of security
  - Lattices: ~ 60,000 bits
  - RSA: ~ 1000 bits

Digital Signatures

- Arguably the most important application of public key cryptography
- Signature lengths for ~ 80 bits of security
  - Lattices: ~ 60,000 bits
  - RSA: ~ 1000 bits
- If we want lattices to be a viable alternative, we **must** make signatures smaller

Digital Signatures

- Arguably the most important application of public key cryptography
- Signature lengths for ~ 80 bits of security
  - Lattices: ~ 60,000 bits
  - RSA: ~ 1000 bits
- If we want lattices to be a viable alternative, we **must** make signatures smaller
  - In my opinion, this, and constructing 'practical' fully-homomorphic encryption are the two most important problems in lattice-based crypto

In this Talk

In this Talk

- A new way to construct lattice-based signature schemes

In this Talk

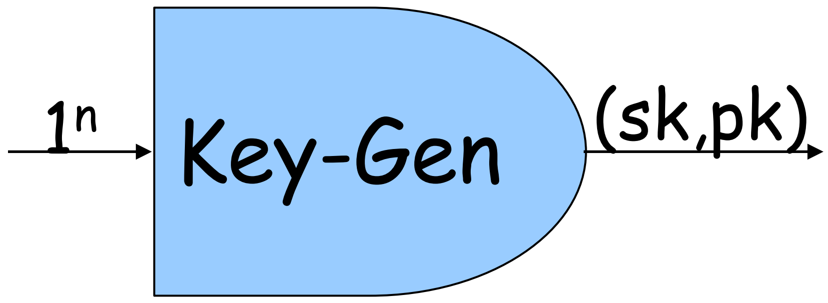
- A new way to construct lattice-based signature schemes
- For ~ 80 bits of security:
  - public key ~ 12,000 bits
  - secret key ~ 1700 bits
  - signature size ~ 9000 bits
  - much faster than RSA/EC signatures

Digital Signature Schemes

Consist of three algorithms: Key-Generate, Sign, and Verify

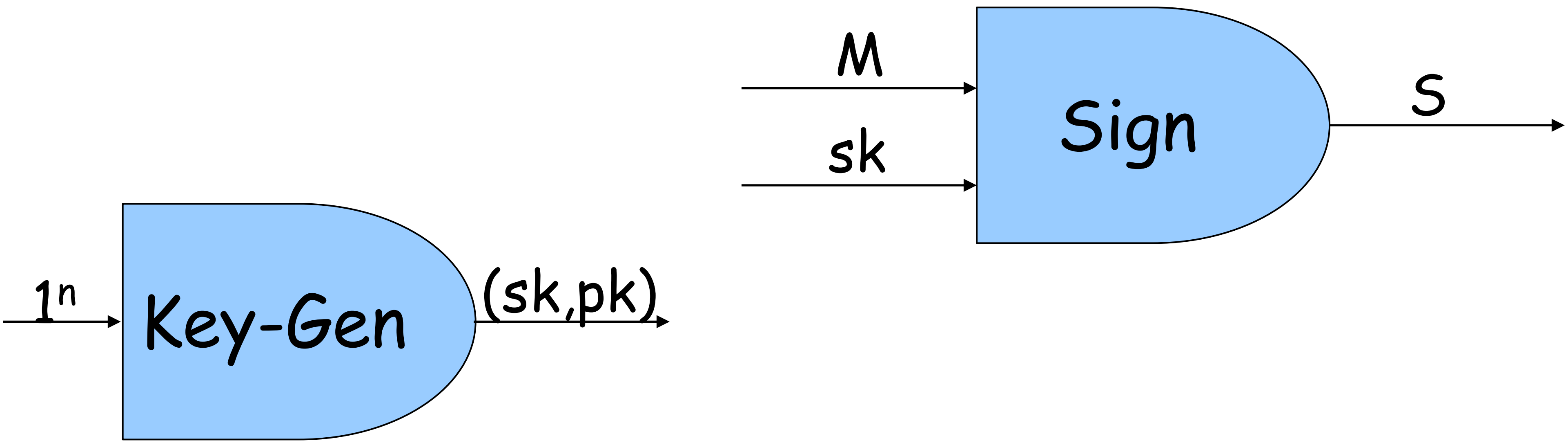
Digital Signature Schemes

Consist of three algorithms: Key-Generate, Sign, and Verify



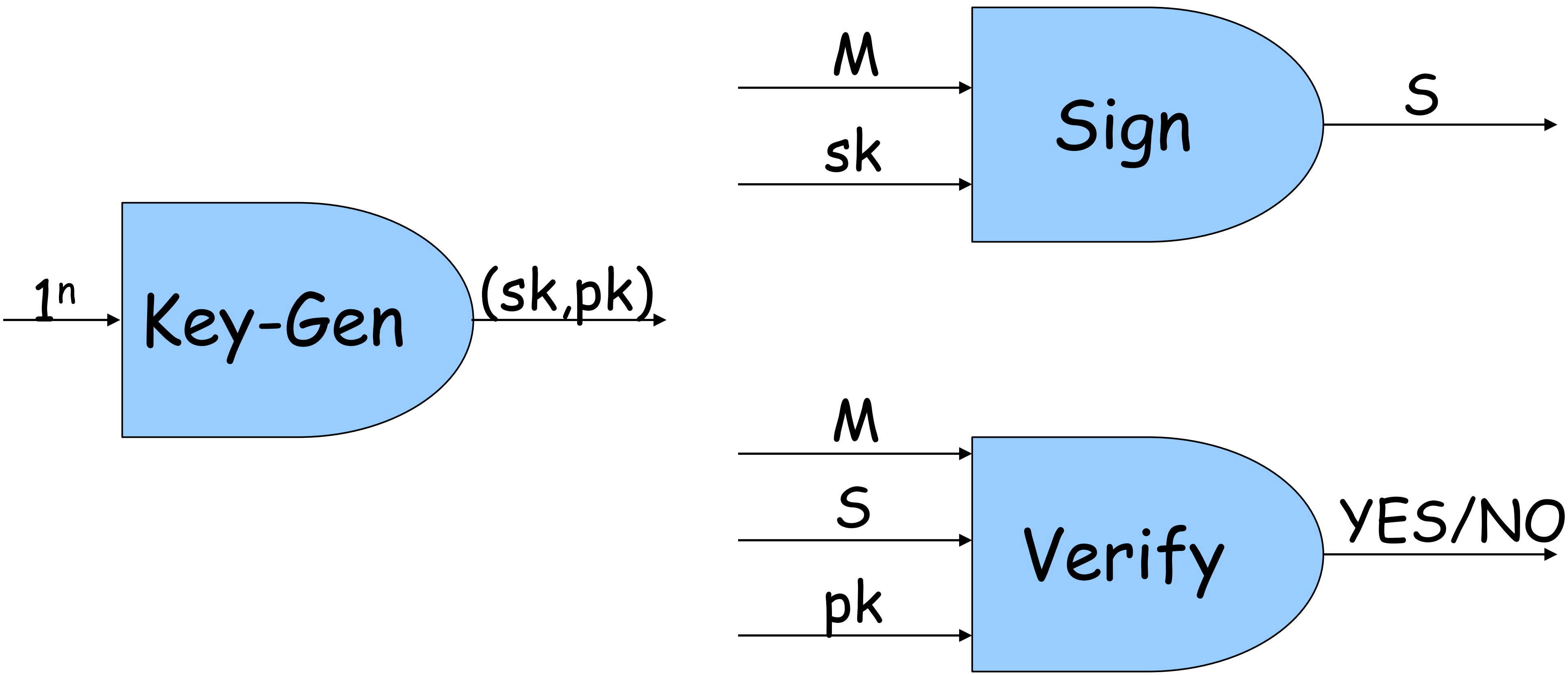
Digital Signature Schemes

Consist of three algorithms: Key-Generate, Sign, and Verify



Digital Signature Schemes

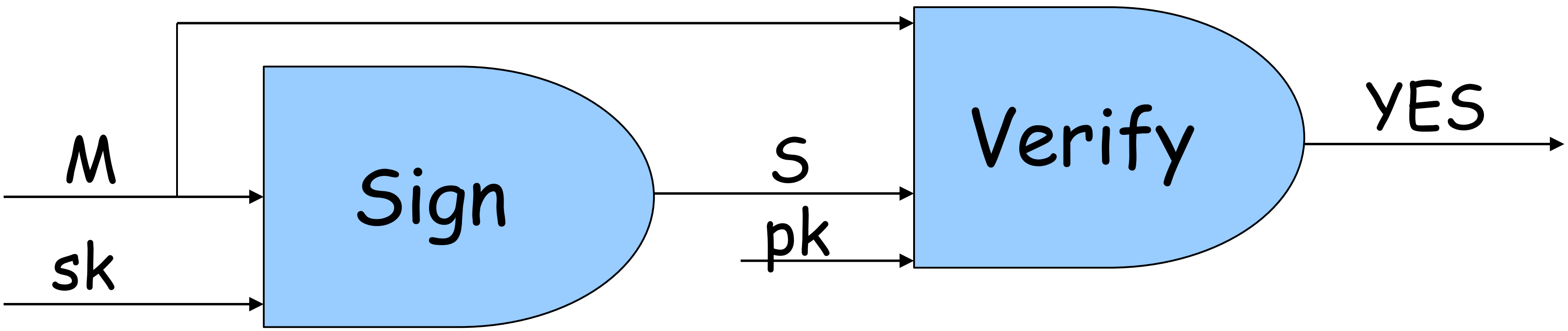
Consist of three algorithms: Key-Generate, Sign, and Verify



Two Properties

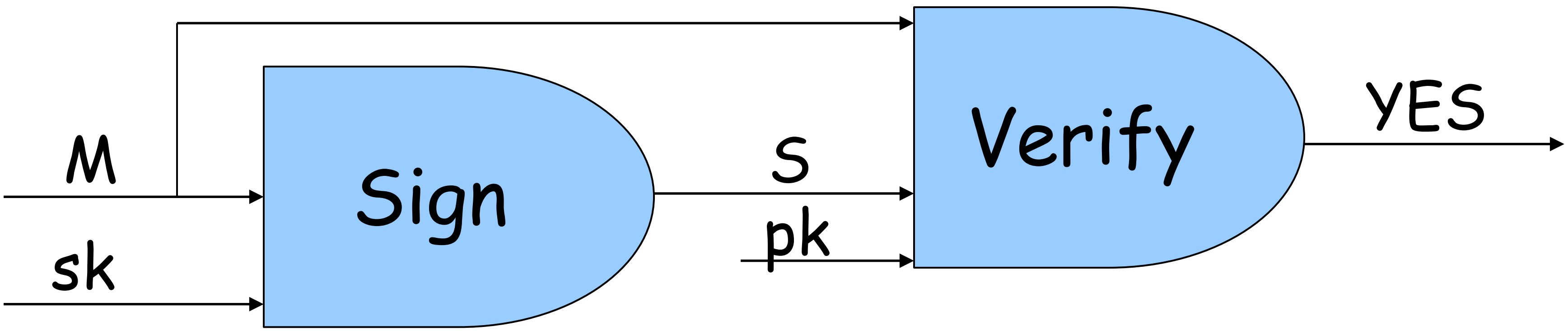
Two Properties

1. Correctness



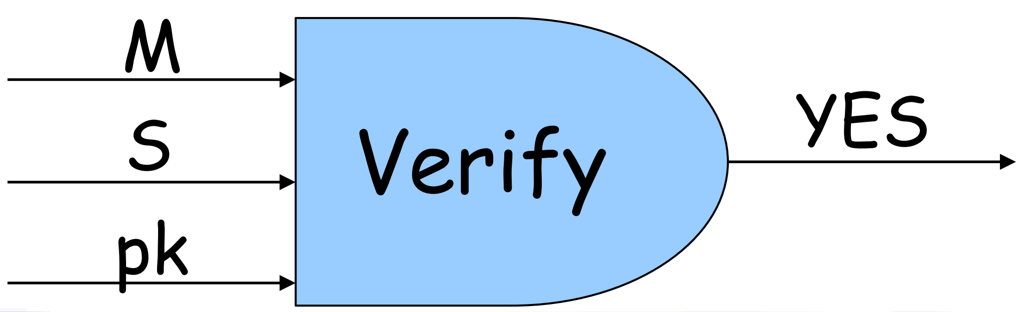
Two Properties

1. Correctness



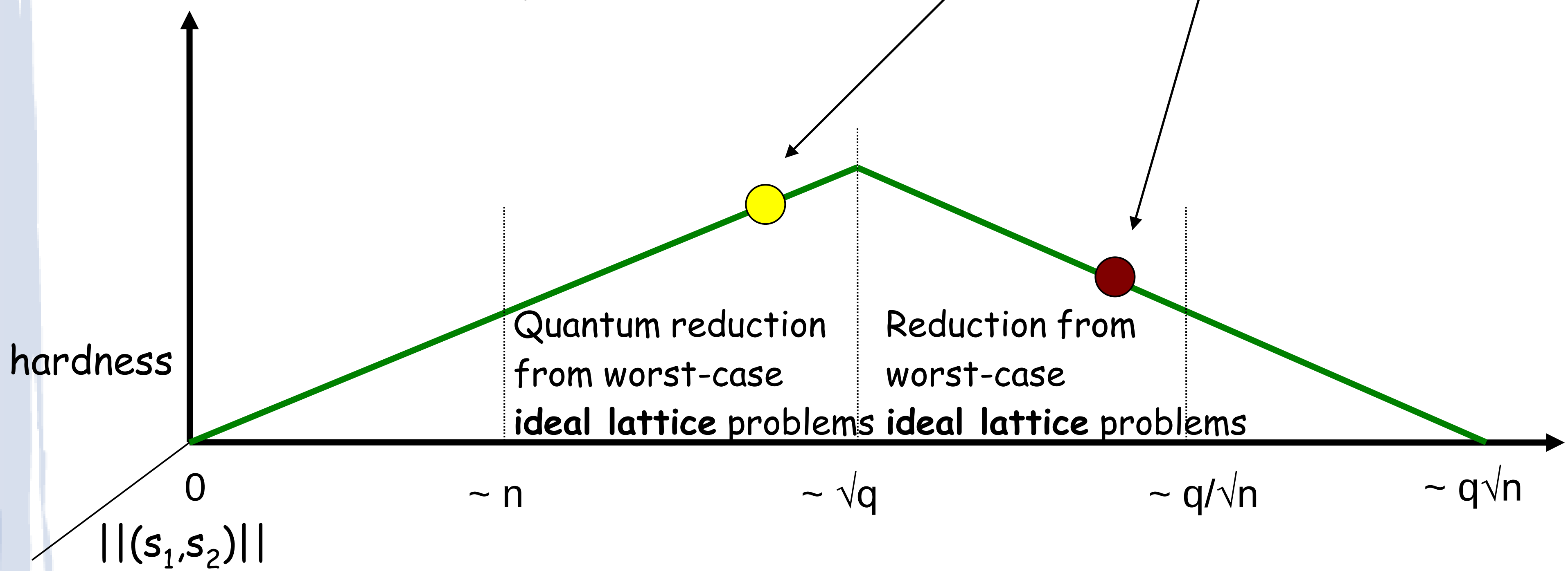
2. Security

Unless M has been signed, cannot find an S such that

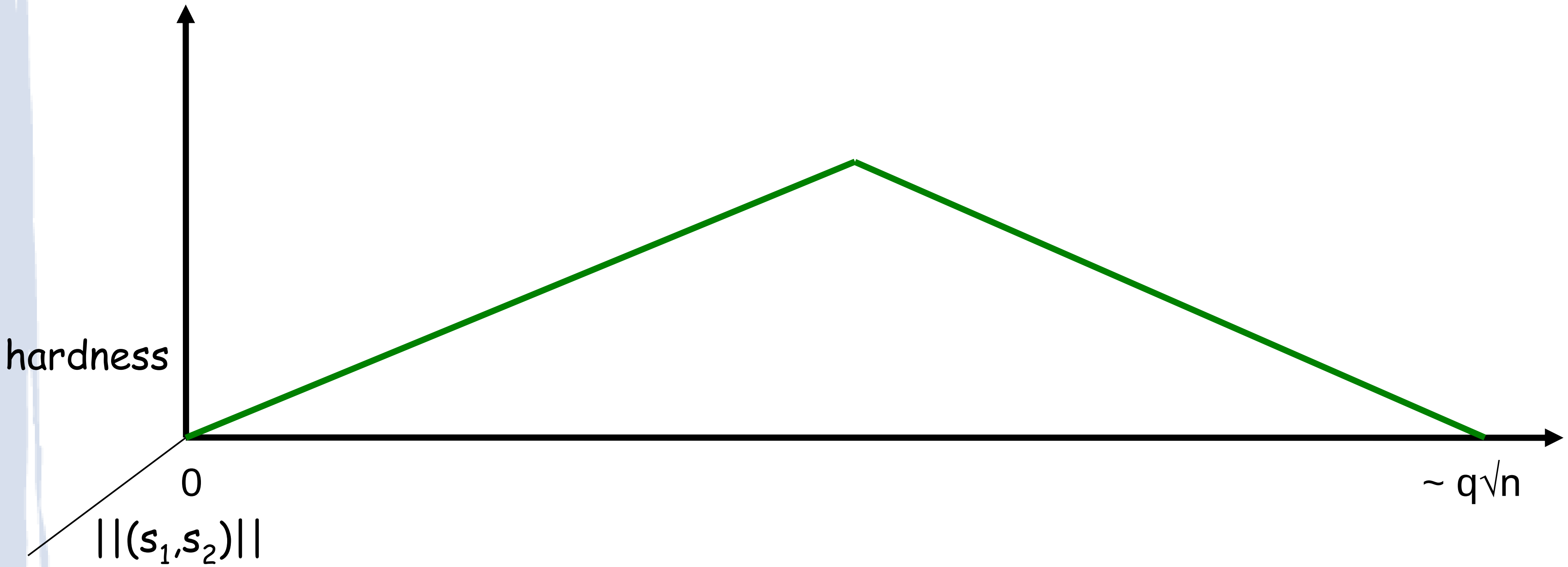


Super High-Level Idea Behind the New Construction

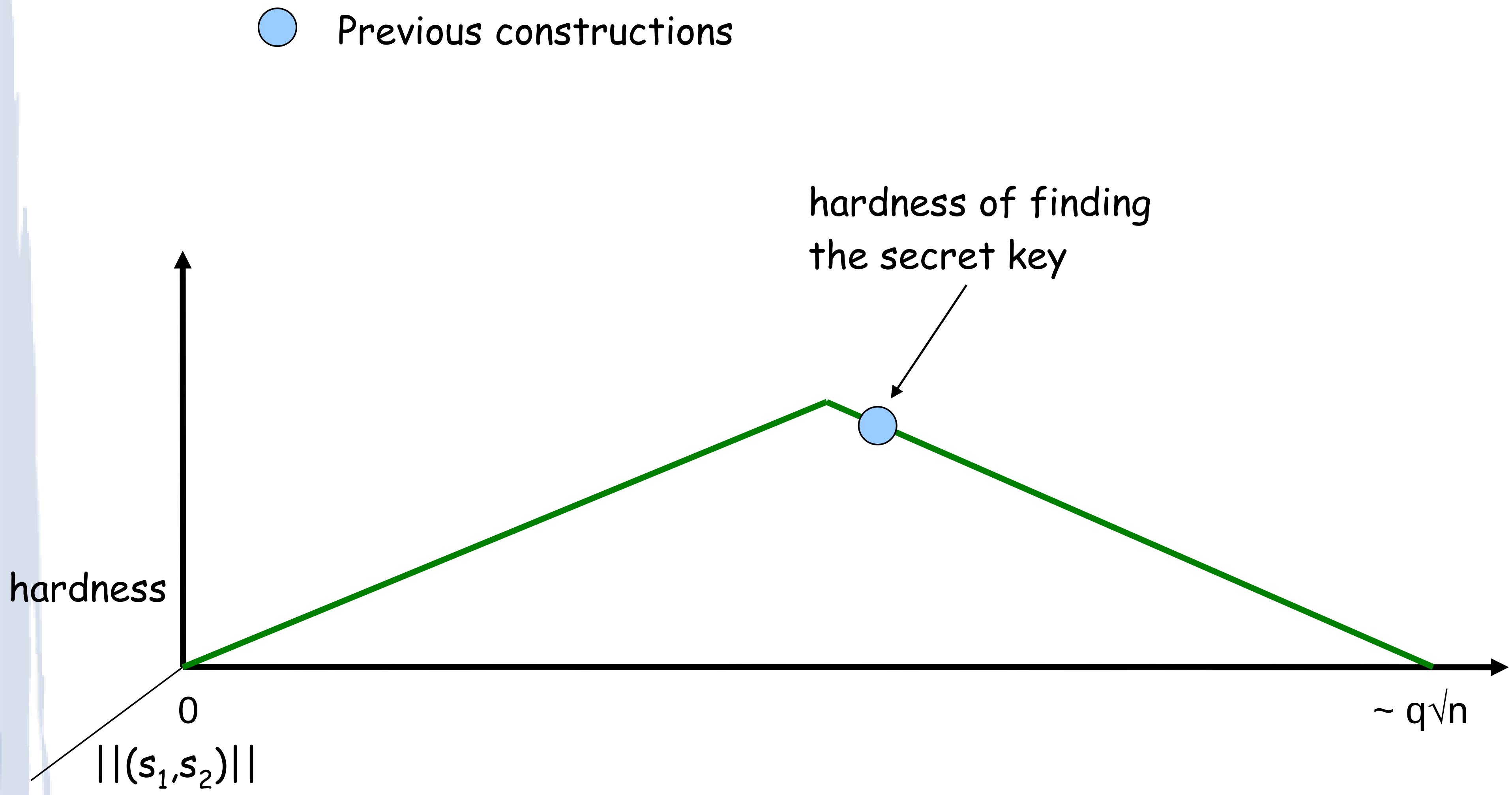
Is it better to have a scheme based on this problem or this problem?
(assuming all other parameters are equal)



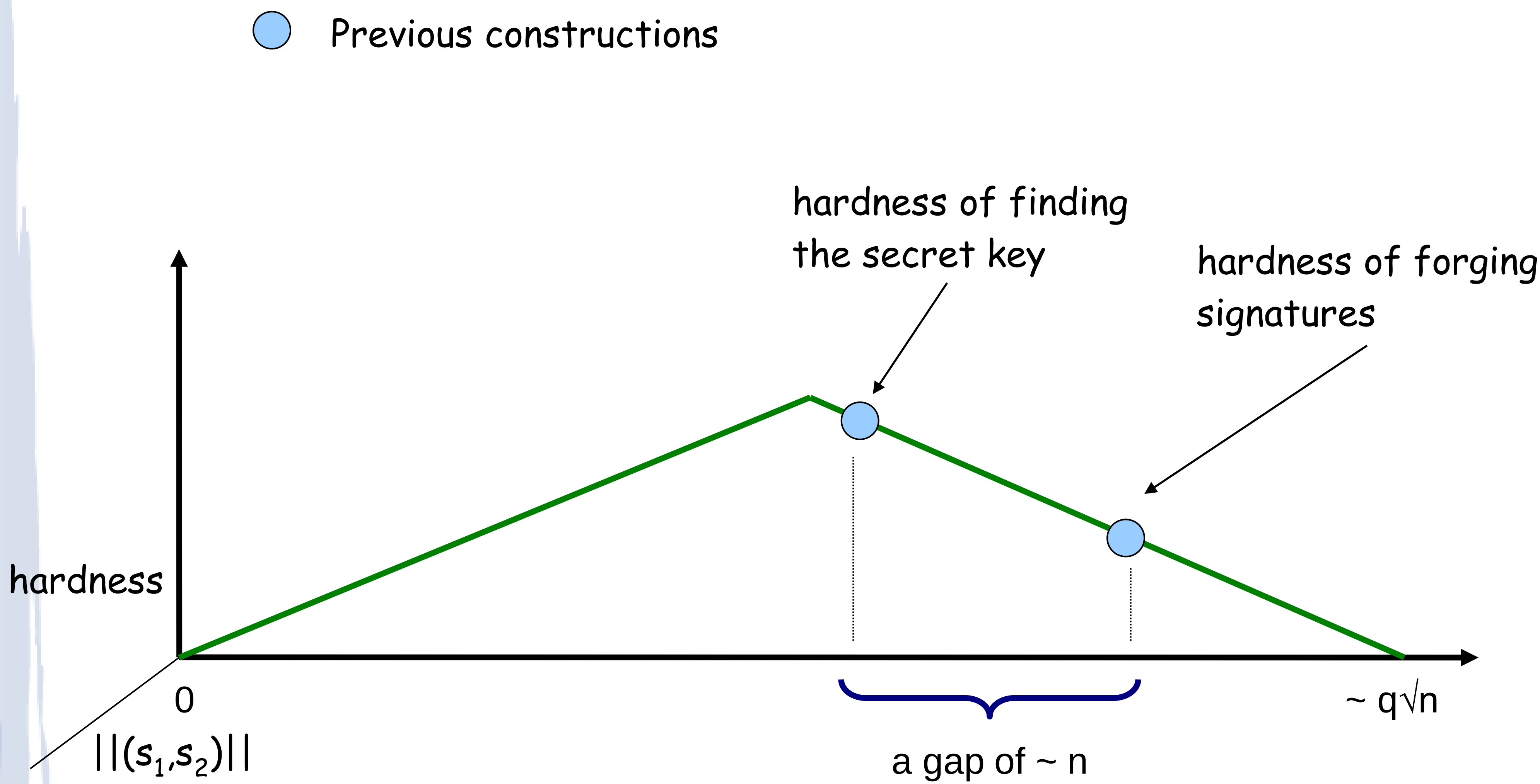
Super High-Level Idea Behind the New Construction



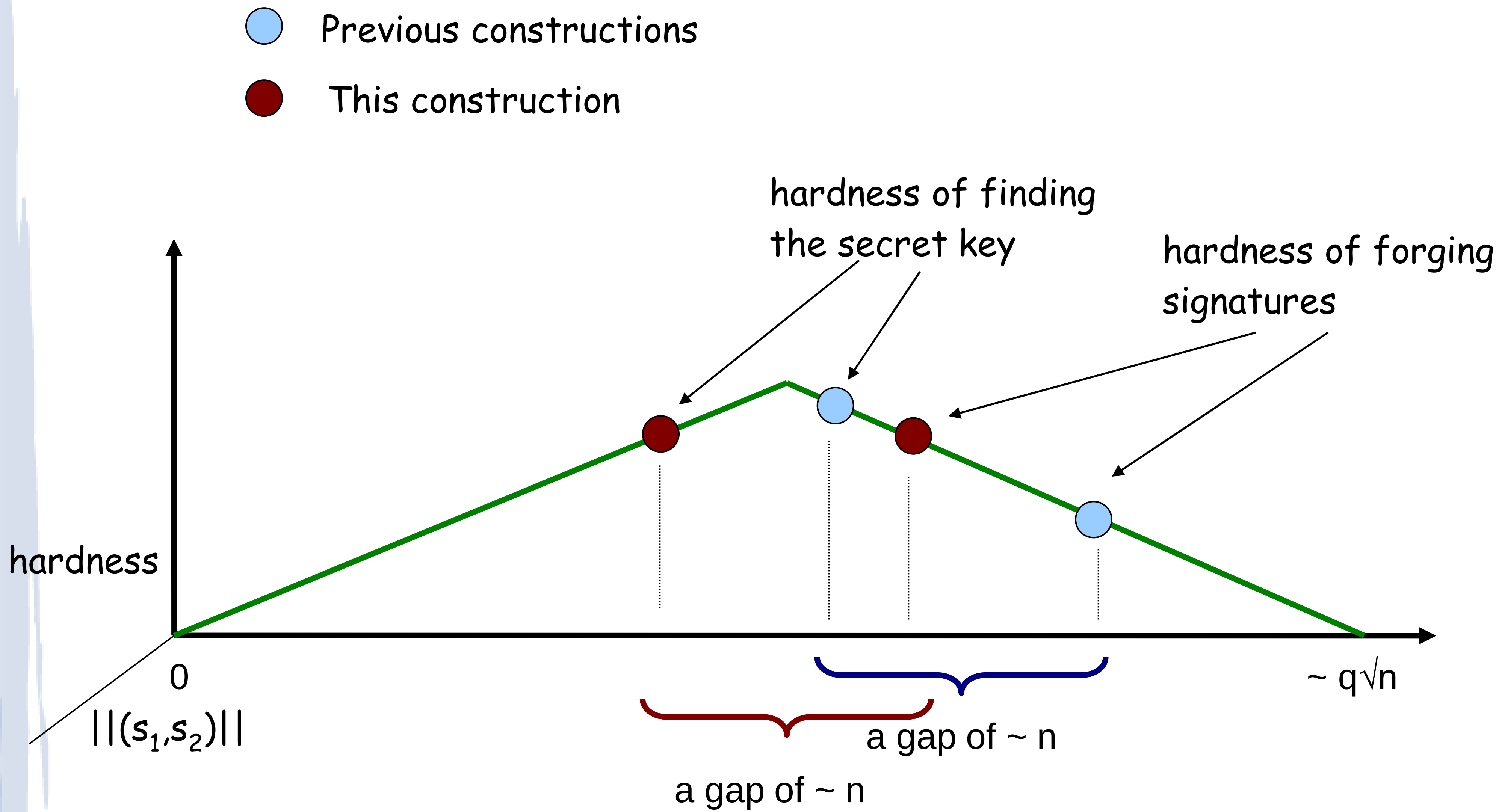
Super High-Level Idea Behind the New Construction



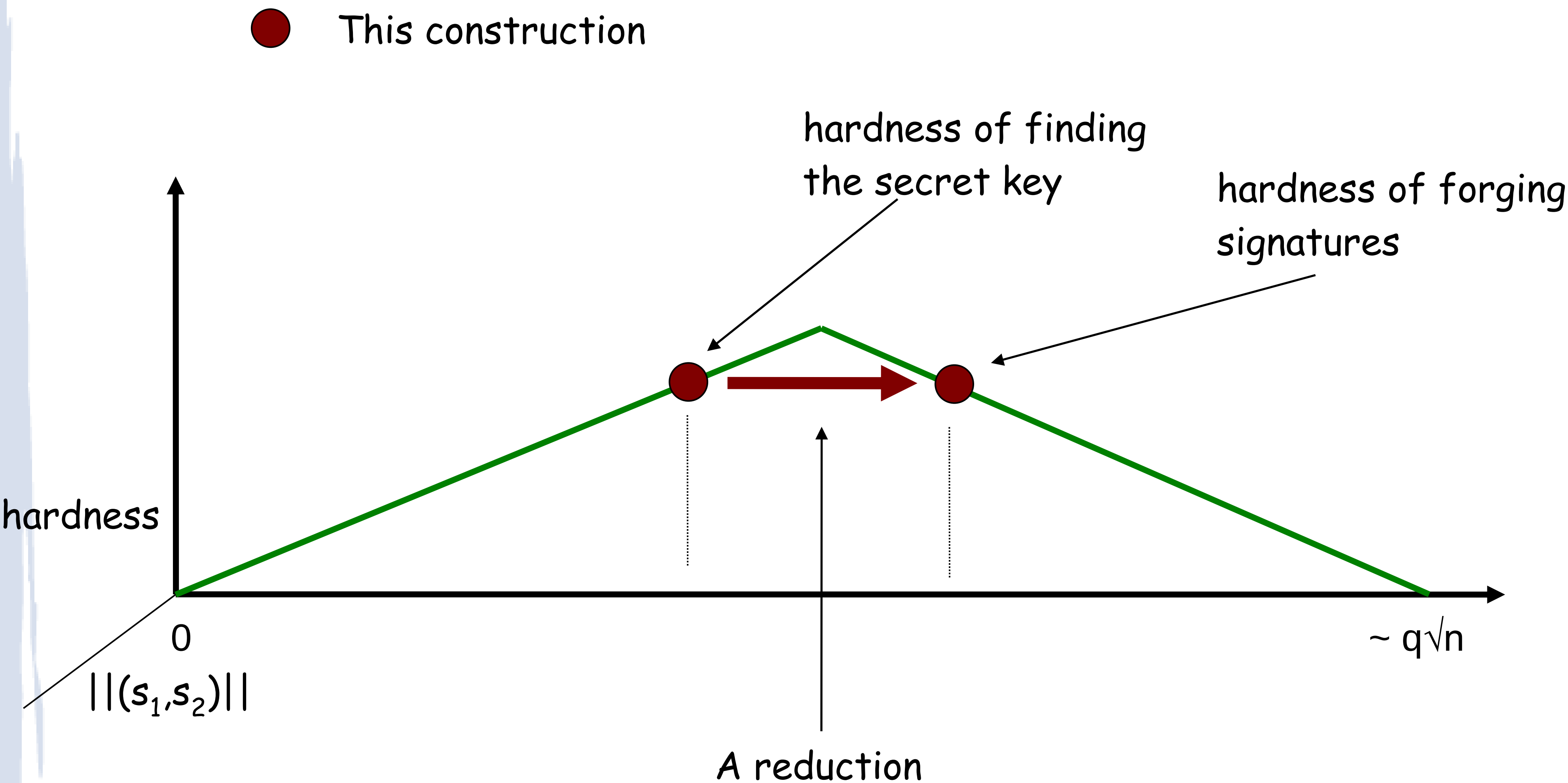
Super High-Level Idea Behind the New Construction



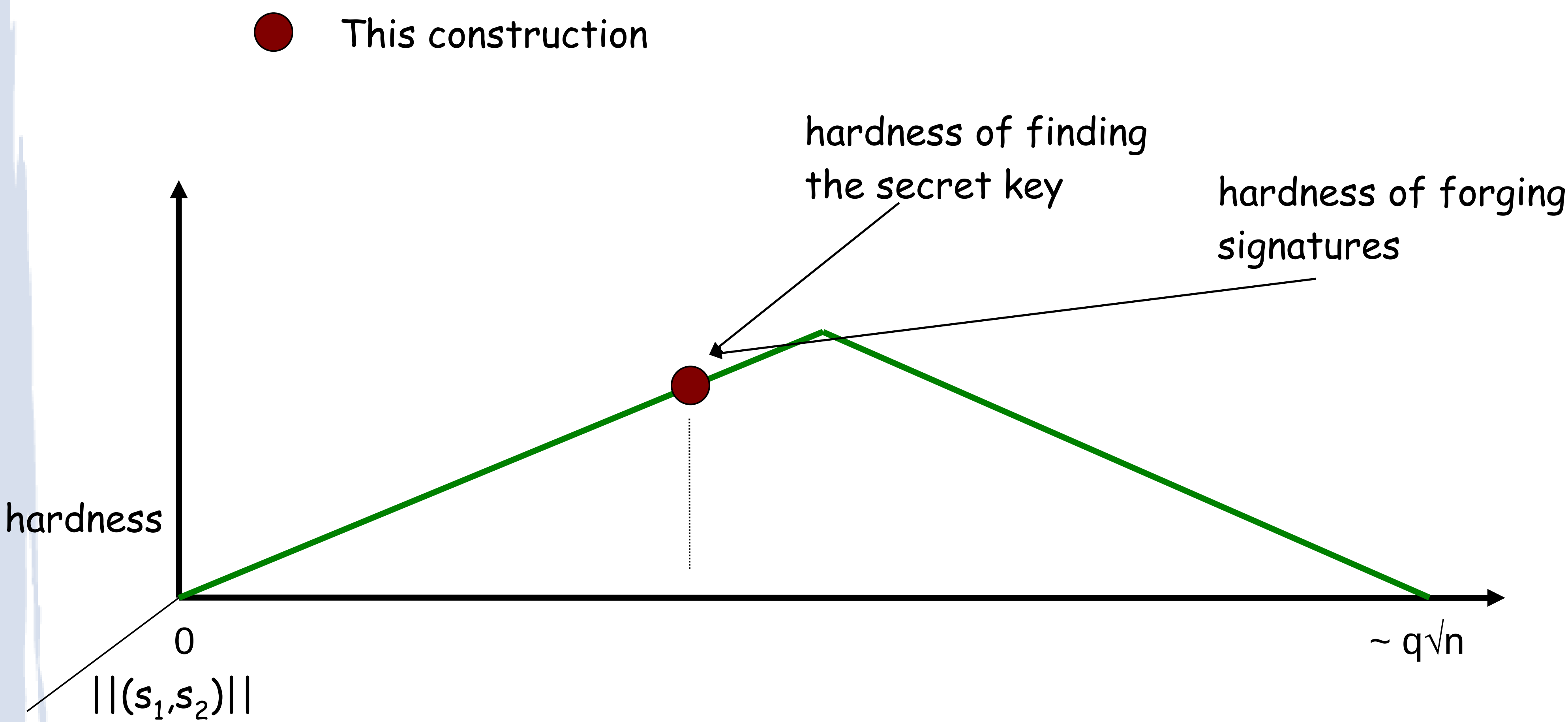
Super High-Level Idea Behind the New Construction



Super High-Level Idea Behind the New Construction



Super High-Level Idea Behind the New Construction



The Ring R

The Ring R

- $R = Z_q[x]/(x^n + 1)$

The Ring R

- $R = Z_q[x]/(x^n + 1)$

  n is a power of 2

  q is a prime ($q = 1 \bmod 2n$)

The Ring R

- $R = Z_q[x]/(x^n + 1)$

  n is a power of 2

  q is a prime ($q = 1 \bmod 2n$)

  Elements in R are polynomials of degree < n

  Coefficients in the range $[-(q-1)/2, (q-1)/2]$

The Ring R

- $R = Z_q[x]/(x^n + 1)$

  n is a power of 2

  q is a prime ($q = 1 \bmod 2n$)

  Elements in R are polynomials of degree < n

  Coefficients in the range $[-(q-1)/2, (q-1)/2]$

- R_k = { polynomials in R with coefficients in the range $[-k,k]$ }

The Compact Knapsack Problem

(The Search Version)

The Compact Knapsack Problem

(The Search Version)

SCK(k):

- pick random a in R
- pick random s_1, s_2 in R_k
- output (a, $b = as_1 + s_2$)

The Compact Knapsack Problem

(The Search Version)

SCK(k):

- pick random a in R
- pick random s_1, s_2 in R_k
- output (a, $b = as_1 + s_2$)

Given (a,b), find s_1, s_2 in R_k such that $as_1+s_2 = b$

(note: there could be more than one solution)

The Compact Knapsack Problem

(The Decision Version)

The Compact Knapsack Problem

(The Decision Version)

DCK(k):

- pick random a,u in R
- pick random c in {0,1}
- pick random s_1, s_2 in R_k
- output (a, $b=as_1 + s_2 + cu$)

The Compact Knapsack Problem

(The Decision Version)

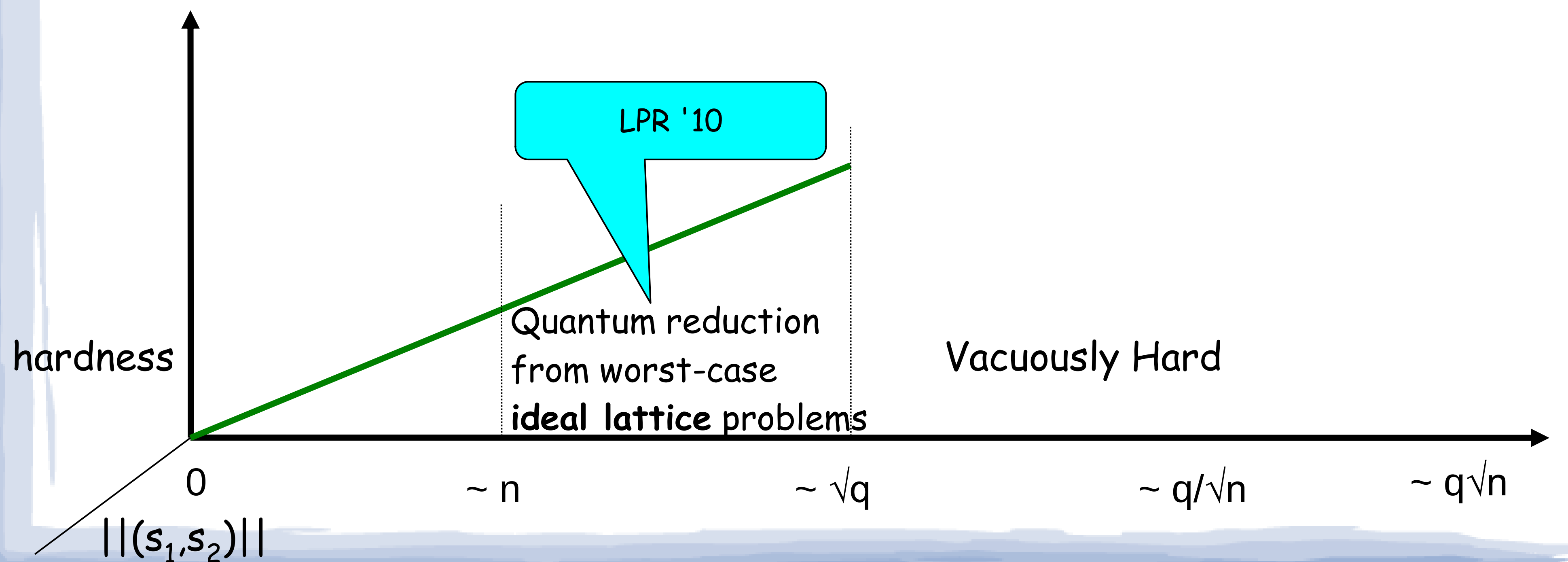
DCK(k):

- pick random a,u in R
- pick random c in {0,1}
- pick random s_1, s_2 in R_k
- output (a, $b=as_1 + s_2 + cu$)

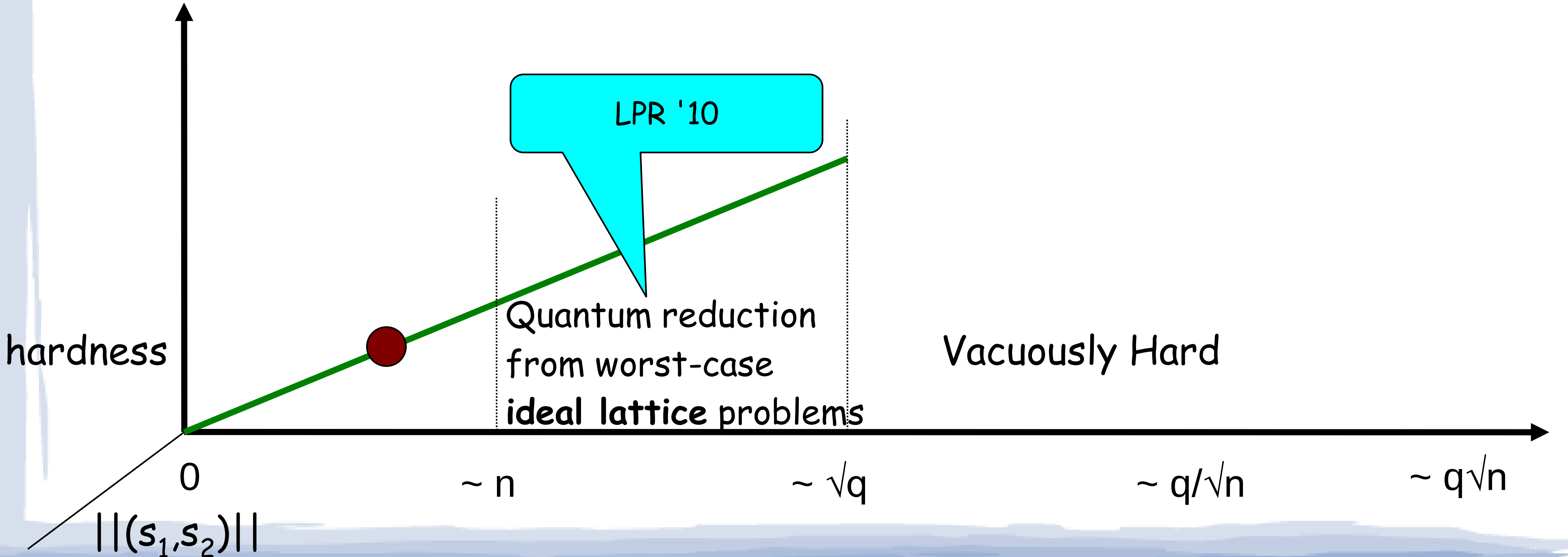
Given (a,b), find c (be correct with probability > 1/2)

- Note: if k is too big, the problem is vacuously hard

Hardness of the Compact Knapsack Problem (Decision Version)



For Added Efficiency …



The Signature Scheme

The Signature Scheme

sk: s_1, s_2 in R_1 pk: a in R, $b=as_1+s_2$

The Signature Scheme

sk: s_1, s_2 in R_1 pk: a in R, $b=as_1+s_2$

sign(m)

1. pick random y_1,y_2 in R_k ($k \sim n$)

The Signature Scheme

sk: s_1, s_2 in R_1 pk: a in R, $b=as_1+s_2$

sign(m)

1. pick random y_1,y_2 in R_k ($k \sim n$)
2. $c = H(ay_1+y_2\ , m)$ Range of H: sparse polynomials in R_1 (at most 32 non-zero elements)

The Signature Scheme

sk: s_1, s_2 in R_1 pk: a in R, $b=as_1+s_2$

sign(m)

1. pick random y_1, y_2 in R_k ($k \sim n$)
2. $c = H(ay_1+y_2\ ,\ m)$ Range of H: sparse polynomials in R_1 (at most 32 non-zero elements)
3. $z_1=cs_1+y_1\ ,\ z_2=cs_2+y_2$

The Signature Scheme

sk: s_1, s_2 in R_1 pk: a in R, $b=as_1+s_2$

sign(m)

1. pick random y_1, y_2 in R_k ($k \sim n$)
2. $c = H(ay_1+y_2\ , m)$ Range of H: sparse polynomials in R_1 (at most 32 non-zero elements)
3. $z_1=cs_1+y_1\ ,\ z_2=cs_2+y_2$
4. if z_1, z_2 are not in R_{k-32} , go back to step 1

The Signature Scheme

sk: s_1, s_2 in R_1 pk: a in R, $b=as_1+s_2$

sign(m)

1. pick random y_1, y_2 in R_k ($k \sim n$)
2. $c = H(ay_1+y_2\ ,\ m)$ Range of H: sparse polynomials in R_1 (at most 32 non-zero elements)
3. $z_1=cs_1+y_1\ ,\ z_2=cs_2+y_2$
4. if z_1, z_2 are not in R_{k-32} , go back to step 1
5. output (z_1, z_2, c) Happens with probability $\sim (1-32/k)^{2n}$

The Signature Scheme

sk: s_1, s_2 in R_1 pk: a in R, $b=as_1+s_2$

sign(m)

1. pick random y_1,y_2 in R_k ($k \sim n$)
2. $c = H(ay_1+y_2 , m)$ Range of H: sparse polynomials in R_1 (at most 32 non-zero elements)
3. $z_1=cs_1+y_1$, $z_2=cs_2+y_2$
4. if z_1,z_2 are not in R_{k-32} , go back to step 1
5. output (z_1, z_2, c) Happens with probability $\sim (1\text{-}32/k)^{2n}$

verify(z_1,z_2,c)

check that z_1,z_2 are in R_{k-32} and $c=H(az_1 + z_2 - bc, m)$

The Signature Scheme

sk: s_1, s_2 in R_1 pk: a in R, $b=as_1+s_2$

sign(m)

1. pick random y_1,y_2 in R_k ($k \sim n$)
2. $c = H(ay_1+y_2 , m)$
3. $z_1=cs_1+y_1$, $z_2=cs_2+y_2$
4. if z_1,z_2 are not in R_{k-32} , go back to step 1
5. output (z_1, z_2, c)

signature size ~ nlog(2k)+nlog(2k)+160

verify(z_1,z_2,c)

check that z_1,z_2 are in R_{k-32} and $c=H(az_1 + z_2 - bc, m)$

The Signature Scheme
(improved version)

sk: s_1, s_2 in R_1 pk: a in R, $b=as_1+s_2$

sign(m)

H only acts on the "high order bits"

1. pick random y_1,y_2 in R_k ($k \sim n$)
2. $c = H(ay_1+y_2 , m)$
3. $z_1=cs_1+y_1$, $z_2=cs_2+y_2$

can "compress" z_2

4. if z_1,z_2 are not in R_{k-32} , go back to step 1
5. output (z_1, z_2, c)

verify(z_1,z_2,c)

check that z_1,z_2 are in R_{k-32} and $c=H(az_1 + z_2 - bc, m)$

The Signature Scheme

(improved version)

sk: s_1, s_2 in R_1 pk: a in R, $b=as_1+s_2$

sign(m)

H only acts on the "high order bits"

1. pick random y_1,y_2 in R_k ($k \sim n$)
2. $c = H(ay_1+y_2, m)$
3. $z_1=cs_1+y_1$, $z_2=cs_2+y_2$
4. if z_1,z_2 are not in R_{k-32}, go back to step 1
5. output (z_1, z_2, c)

can "compress" z_2

signature size ~ $n\log(2k)+2n+160$

verify(z_1,z_2,c)

check that z_1,z_2 are in R_{k-32} and $c=H(az_1 + z_2 - bc, m)$

The Signature Scheme

sk: s_1, s_2 in R_1 pk: a in R, $b=as_1+s_2$

sign(m)

1. pick random y_1,y_2 in R_k ($k \sim n$)
2. $c = H(ay_1+y_2 , m)$
3. $z_1=cs_1+y_1$, $z_2=cs_2+y_2$
4. if z_1,z_2 are not in R_{k-32} , go back to step 1
5. output (z_1, z_2, c)

verify(z_1,z_2,c)

check that z_1,z_2 are in R_{k-32} and $c=H(az_1 + z_2 - bc, m)$

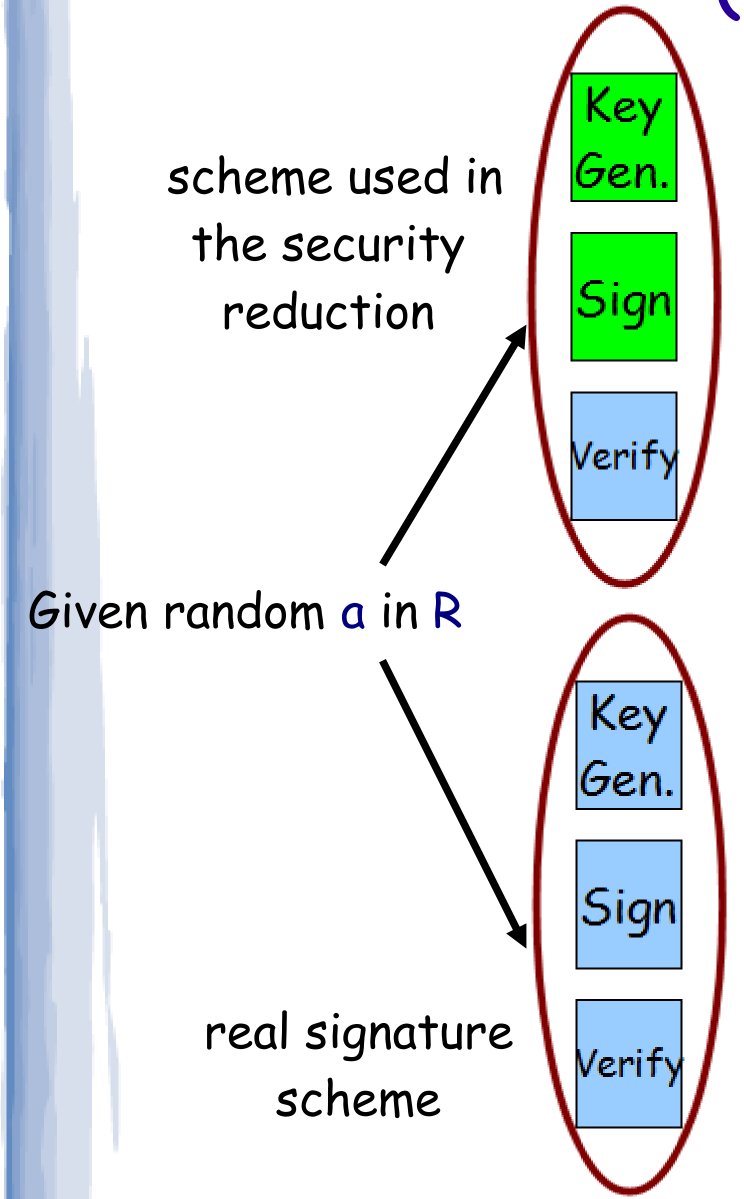
Security Proof

(High Level Idea)

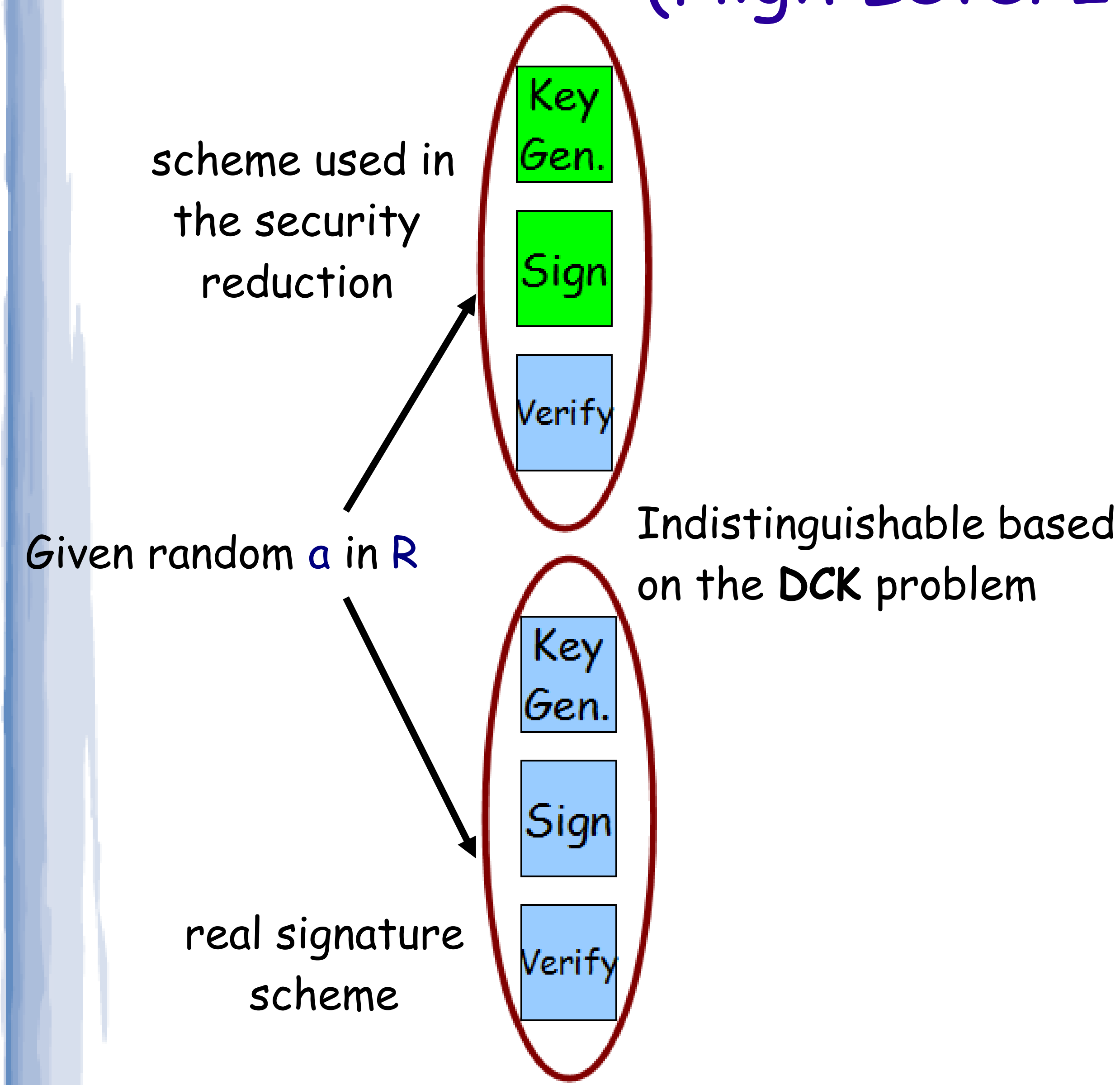
Security Proof
(High Level Idea)

Given random a in R

Security Proof
(High Level Idea)

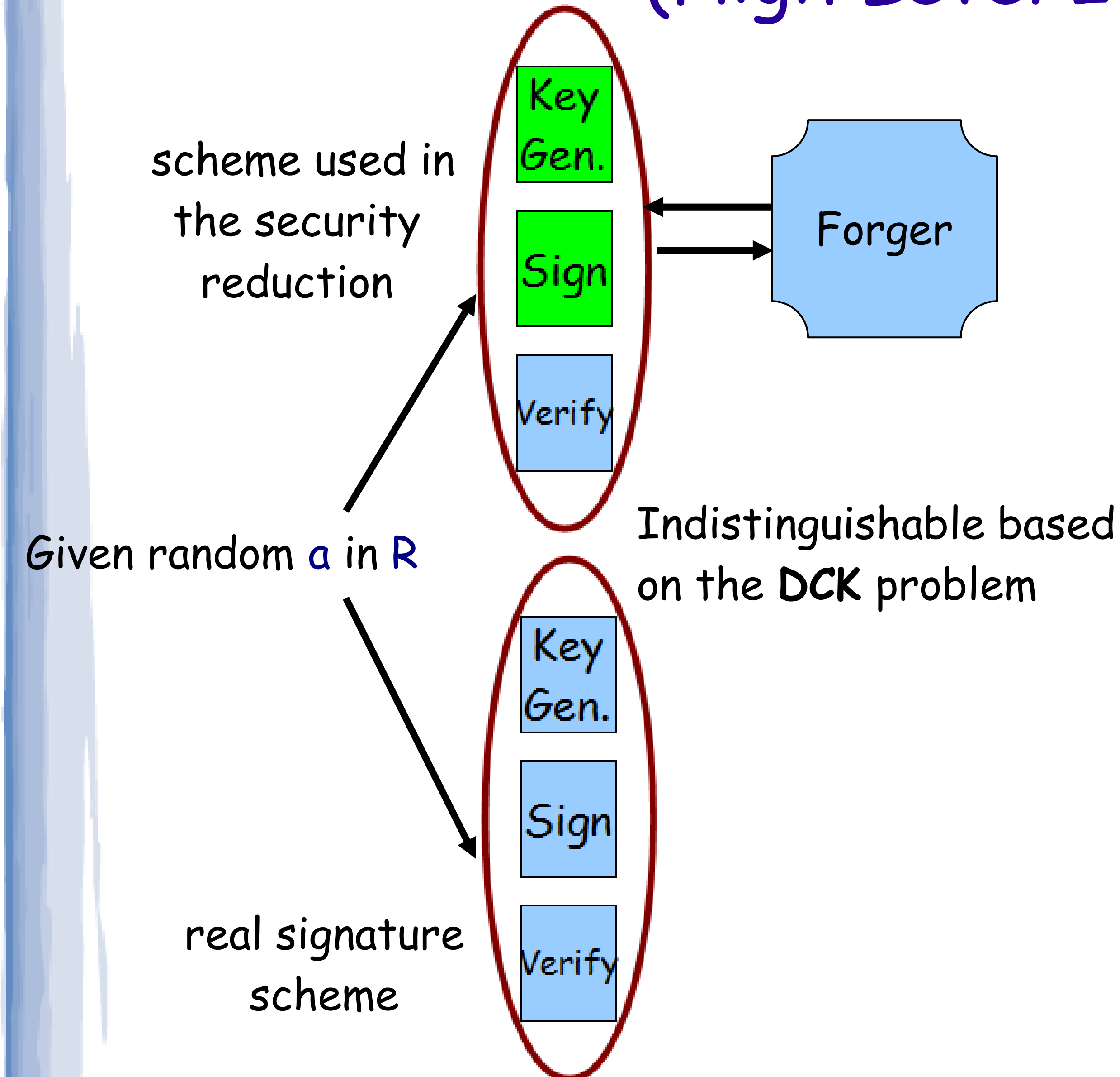


Security Proof
(High Level Idea)

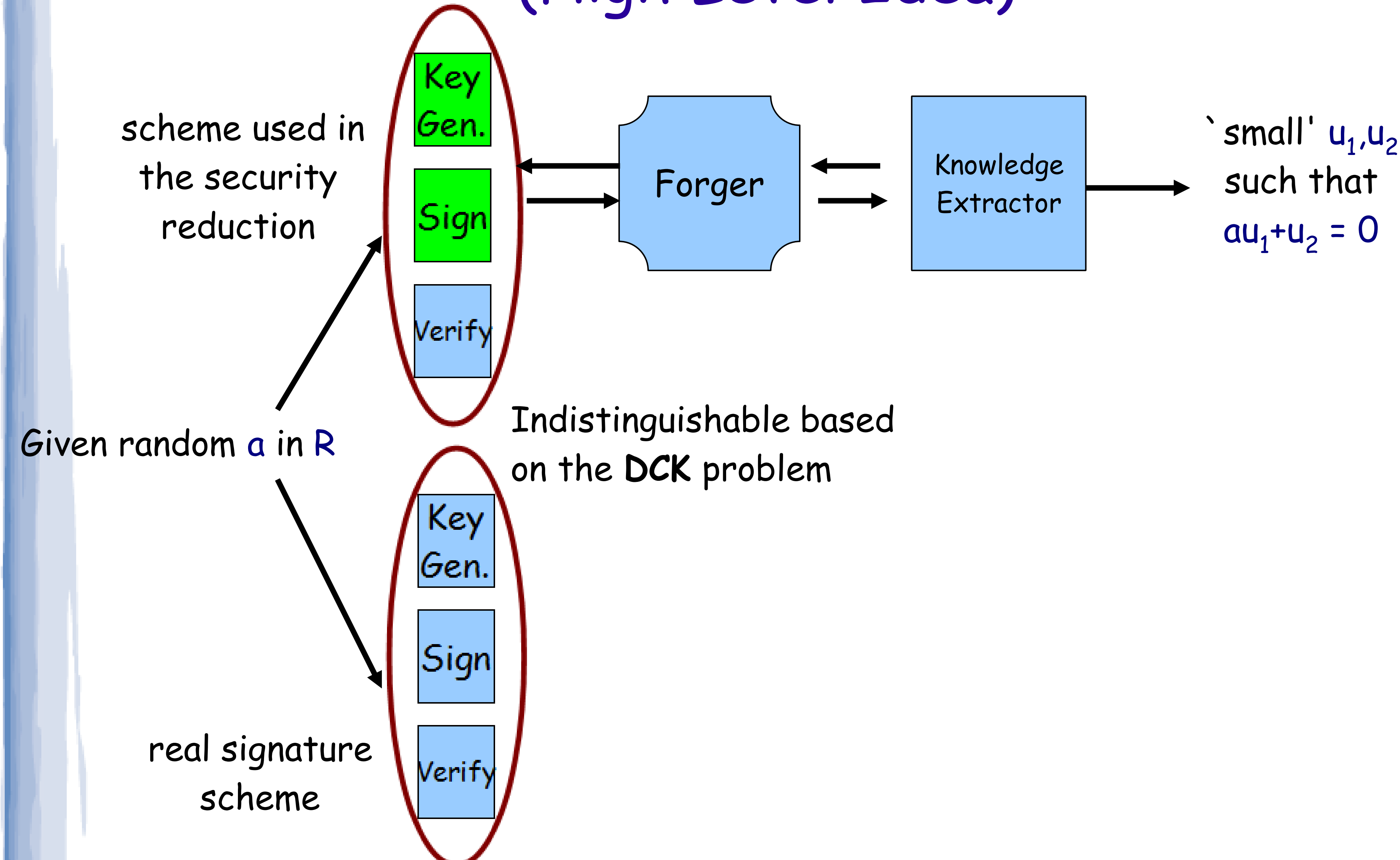


Security Proof

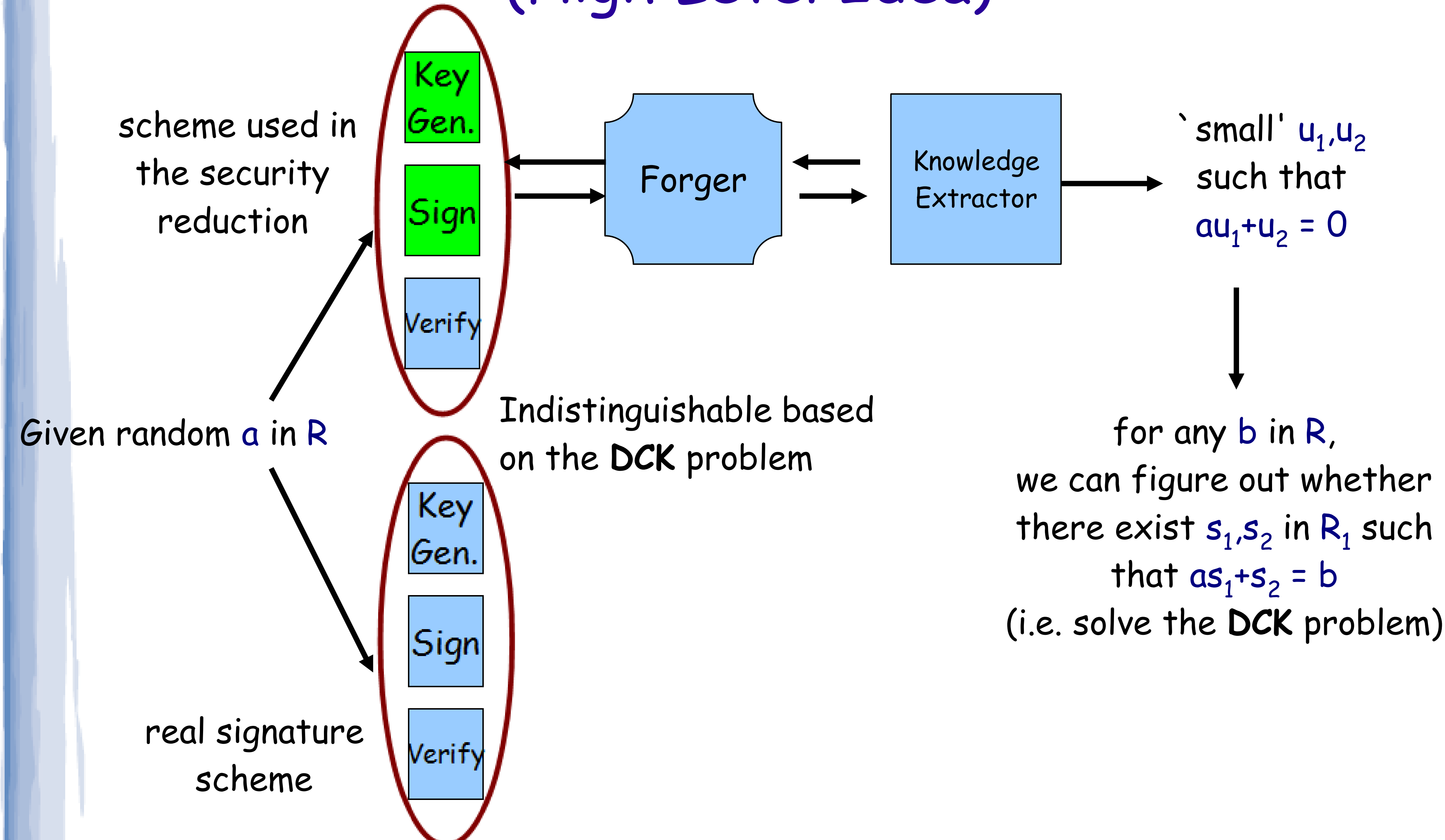
(High Level Idea)



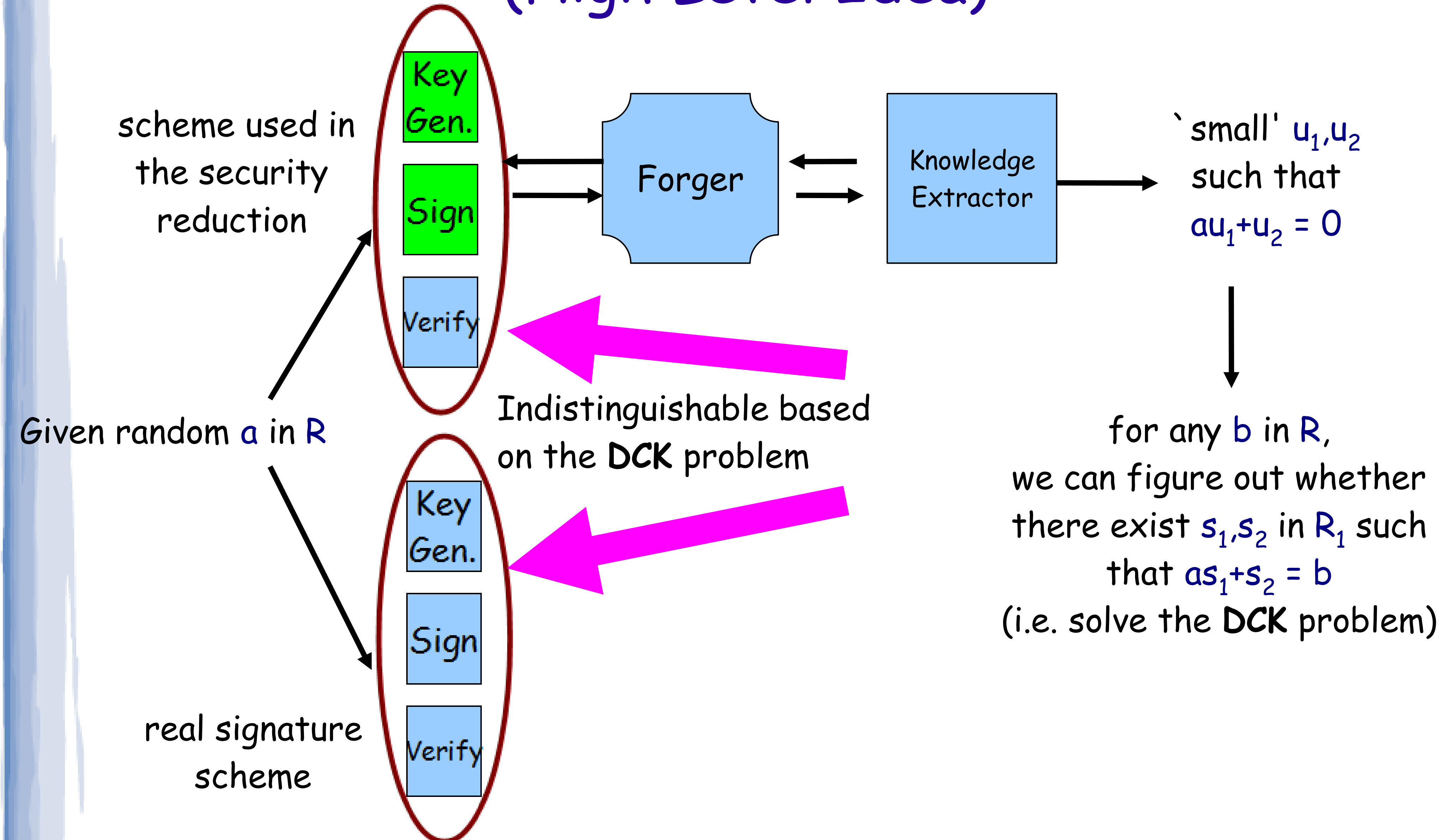
Security Proof
(High Level Idea)



Security Proof
(High Level Idea)



Security Proof
(High Level Idea)



Security Proof

sk: s_1, s_2 in R_1 pk: a in R, $b=as_1+s_2$

sign(m)

1. pick random y_1,y_2 in R_k ($k \sim n$)
2. $c = H(ay_1+y_2 , m)$
3. $z_1=cs_1+y_1$, $z_2=cs_2+y_2$
4. if z_1,z_2 are not in R_{k-32} , go back to step 1
5. output (z_1, z_2, c)

verify(z_1,z_2,c)
check that z_1,z_2 are in R_{k-32} and $c=H(az_1 + z_2 - bc, m)$

Security Proof

sk: s_1, s_2 in R_1 pk: a in R, $b=as_1+s_2$

sign(m)

1. pick random y_1,y_2 in R_k ($k \sim n$)
2. $c = H(ay_1+y_2, m)$
3. $z_1=cs_1+y_1$, $z_2=cs_2+y_2$
4. if z_1,z_2 are not in R_{k-32}, go back to step 1
5. output (z_1, z_2, c)

sk: s_1, s_2 in R_1 pk: a in R, $b=as_1+s_2$

sign(m)

Pick random c in Range(H)

Pick random z_1,z_2 in R_{k-32}

Program $H(az_1+z_2 - bc, m) = c$

output (z_1, z_2, c)

verify(z_1,z_2,c)

check that z_1,z_2 are in R_{k-32} and $c=H(az_1 + z_2 - bc, m)$

Security Proof

sk: s_1, s_2 in R_1 pk: a in R, $b=as_1+s_2$

sign(m)

1. Pick random c in Range(H)
2. Pick random z_1,z_2 in R_{k-32}
3. Program $H(az_1+z_2 - bc, m) = c$
4. output (z_1, z_2, c)

verify(z_1,z_2,c)
check that z_1,z_2 are in R_{k-32} and $c=H(az_1 + z_2 - bc, m)$

Security Proof

sk: s_1, s_2 in R_1 pk: a in R, $b=as_1+s_2$

sign(m)

1. Pick random c in Range(H)
2. Pick random z_1,z_2 in R_{k-32}
3. Program $H(az_1+z_2 - bc, m) = c$
4. output (z_1, z_2, c)

sk: s_1, s_2 in $R_{k'}$ pk: a in R, $b=as_1+s_2$

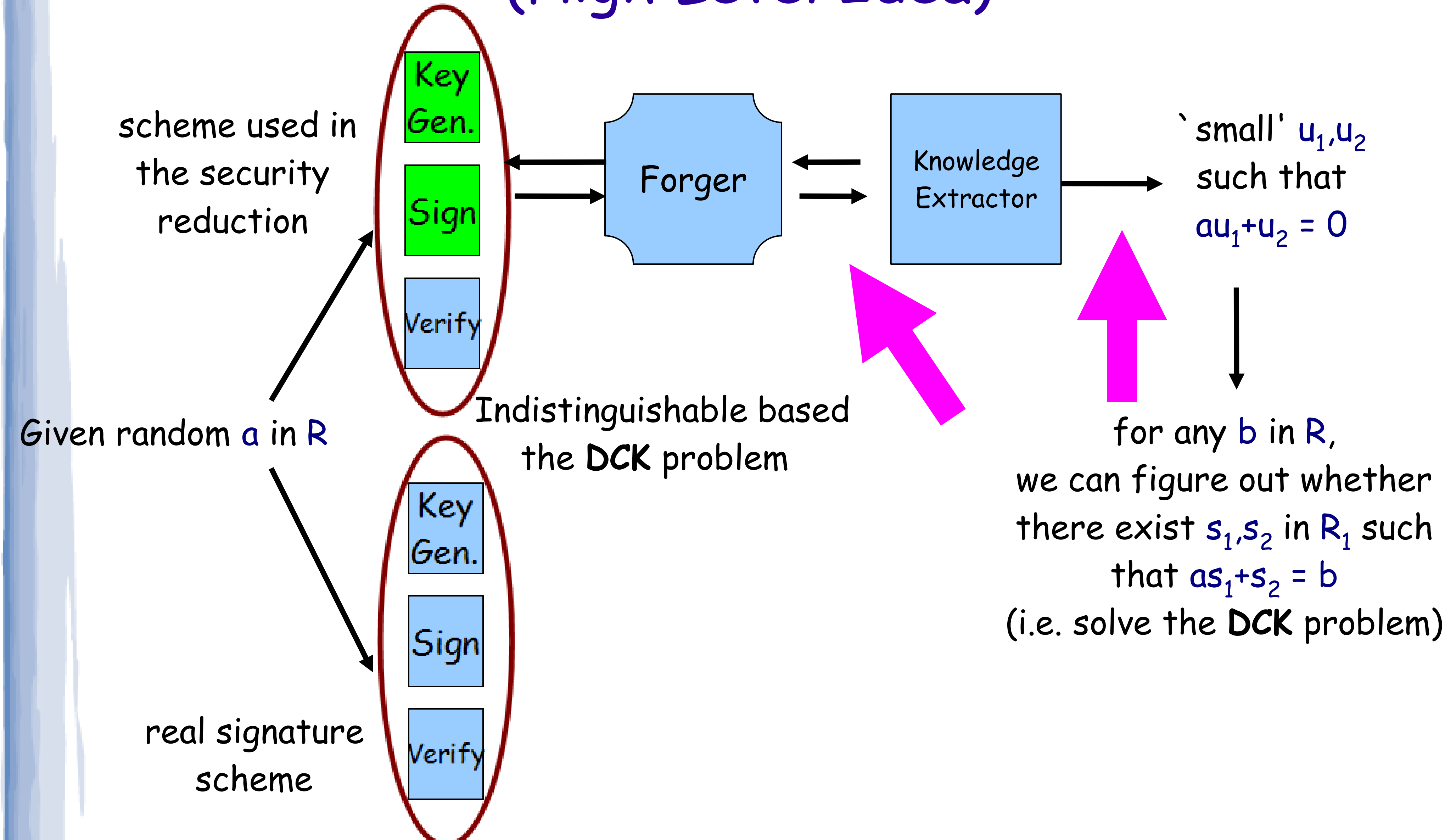
sign(m)

1. Pick random c in Range(H)
2. Pick random z_1,z_2 in R_{k-32}
3. Program $H(az_1+z_2 - bc, m) = c$
4. output (z_1, z_2, c)

verify(z_1,z_2,c)

check that z_1,z_2 are in R_{k-32} and $c=H(az_1 + z_2 - bc, m)$

Security Proof
(High Level Idea)



Security Proof

sk: s_1, s_2 in R_k, pk: a in R, $b=as_1+s_2$

sign(m)

1. Pick random c in Range(H)
2. Pick random z_1,z_2 in R_{k-32}
3. Program $H(az_1+z_2 - bc, m) = c$
4. output (z_1, z_2, c)

verify(z_1,z_2,c)
check that z_1,z_2 are in R_{k-32} and $c=H(az_1 + z_2 - bc, m)$

Security Proof

sk: s_1, s_2 in R_k, pk: a in R, $b=as_1+s_2$

sign(m)

1. Pick random c in Range(H)
2. Pick random z_1,z_2 in R_{k-32}
3. Program $H(az_1+z_2 - bc, m) = c$
4. output (z_1, z_2, c)

We can obtain from a forger two signatures of m

(z_1,z_2,c) and (z'_1,z'_2,c')

such that

$az_1+z_2 - bc = az'_1+z'_2 - bc'$

verify(z_1,z_2,c)

check that z_1,z_2 are in R_{k-32} and $c=H(az_1 + z_2 - bc, m)$

Security Proof

sk: s_1, s_2 in R_k, pk: a in R, $b=as_1+s_2$

sign(m)

1. Pick random c in Range(H)
2. Pick random z_1,z_2 in R_{k-32}
3. Program $H(az_1+z_2 - bc, m) = c$
4. output (z_1, z_2, c)

We can obtain from a forger two signatures of m

(z_1,z_2,c) and (z'_1,z'_2,c')

such that

$az_1+z_2 - bc = az'_1+z'_2 - bc'$

Plugging in $b=as_1+s_2$ ...

$$\underbrace{a(z_1-cs_1-z'_1+c's_1)}_{u_1} + \underbrace{(z_2-cs_2-z'_2+c's_2)}_{u_2} = 0$$

verify(z_1,z_2,c)

check that z_1,z_2 are in R_{k-32} and $c=H(az_1 + z_2 - bc, m)$

Security Proof

sk: s_1, s_2 in R_k, pk: a in R, $b=as_1+s_2$

sign(m)

1. Pick random c in Range(H)
2. Pick random z_1,z_2 in R_{k-32}
3. Program $H(az_1+z_2 - bc, m) = c$
4. output (z_1, z_2, c)

We can obtain from a forger
two signatures of m
(z_1,z_2,c) and (z'_1,z'_2,c')
such that
$az_1+z_2 - bc = az'_1+z'_2 - bc'$

Plugging in $b=as_1+s_2$ ...

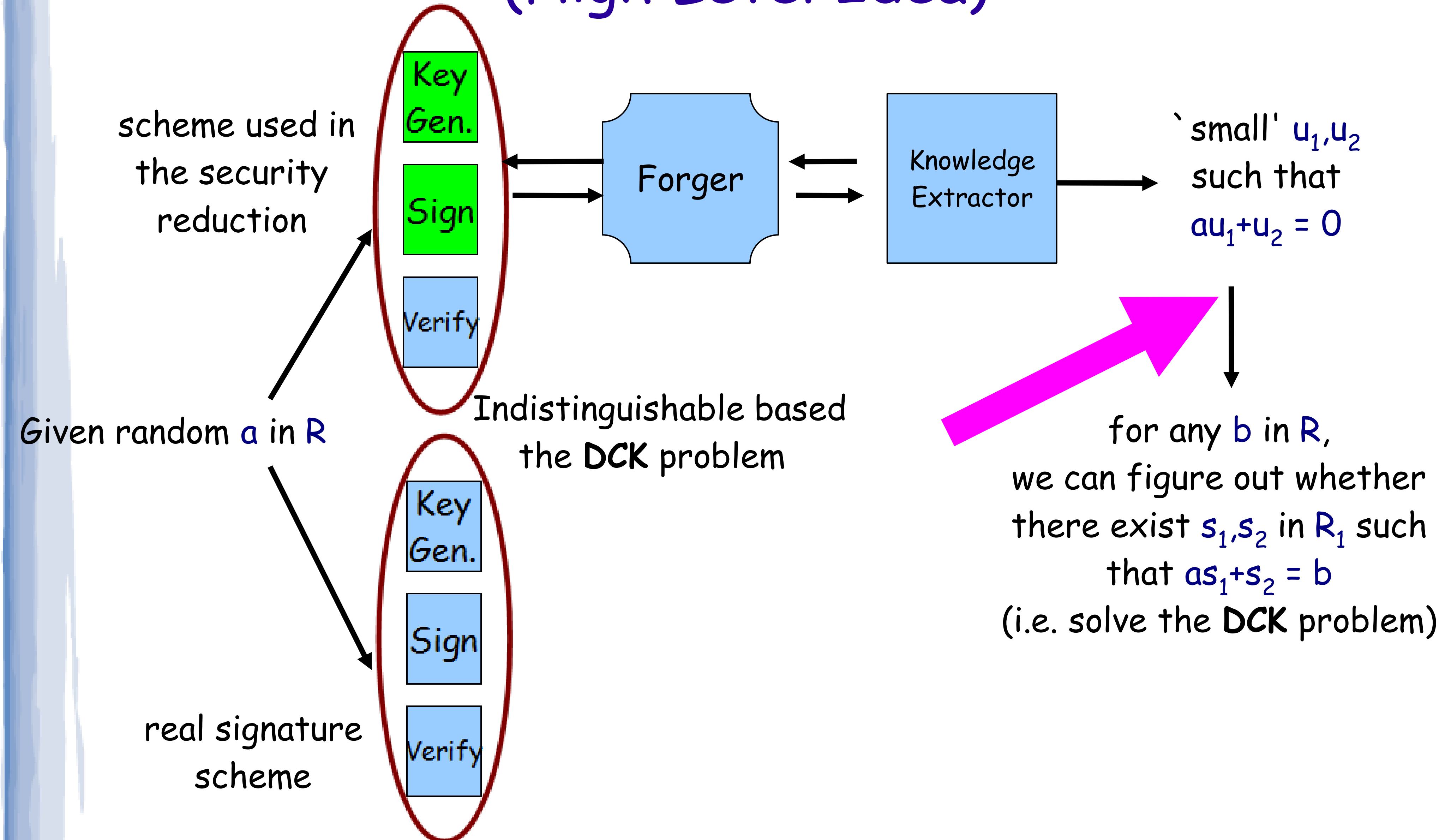
$$\underbrace{a(z_1-cs_1-z'_1+c's_1)}_{u_1} + \underbrace{(z_2-cs_2-z'_2+c's_2)}_{u_2} = 0$$

(Because s_1,s_2 are **not** unique, u_1 and u_2 are not both 0)

verify(z_1,z_2,c)
check that z_1,z_2 are in R_{k-32} and $c=H(az_1 + z_2 - bc, m)$

Security Proof

(High Level Idea)



Security Proof

Security Proof

Given `small' u_1, u_2 such that $au_1+u_2 = 0$, one can solve the DCK problem.

Security Proof

Given `small' u_1, u_2 such that $au_1+u_2 = 0$, one can solve the DCK problem.

Given (a,b), compute u_1b

Security Proof

Given `small' u_1, u_2 such that $au_1+u_2 = 0$, one can solve the DCK problem.

Given (a,b), compute u_1b

- If $b = as1 + s2$ for `small' $s1, s2$, then

  $u1b = u1as1 + u1s2 = -u2s1 + u1s2$ is also `small'

Security Proof

Given `small' u_1, u_2 such that $au_1+u_2 = 0$, one can solve the DCK problem.

Given (a,b), compute u_1b

- If $b={}_a s1_{+s} 2$ for `small' $s1_, s2$, then

  ${}_u1b\ {}_=\ {}_u1as1\ {}_+\ {}_u1s2\ {}_=\ {}_{-u}2s1 + u1s2$ is also `small'

- If b is random, then the coefficients of

  u1b are also random (thus probably `large')

Open Problems

Open Problems

- More efficient signatures?

Open Problems

- More efficient signatures?
- Is the decision assumption necessary?

Open Problems

- More efficient signatures?
- Is the decision assumption necessary?
- Can we construct other efficient lattice-based primitives using this idea?

Open Problems

- More efficient signatures?
- Is the decision assumption necessary?
- Can we construct other efficient lattice-based primitives using this idea?

Thank You!