Smaller decoding exponents: ball-collision decoding

D. J. Bernstein
University of Illinois at Chicago

Joint work with:

Tanja Lange
Technische Universiteit Eindhoven

Christiane Peters
University of Illinois at Chicago

Context: speed

What is the fastest
public-key encryption system?

Context: speed

What is the fastest
public-key encryption system?

RSA-1024 is quite fast.

Context: speed

What is the fastest public-key encryption system?

RSA-1024 is quite fast.
RSA-512 is faster.

Context: speed

What is the fastest
public-key encryption system?

RSA-1024 is quite fast.
RSA-512 is faster.
RSA-256 is even faster.

Context: speed

What is the fastest public-key encryption system?

RSA-1024 is quite fast.
RSA-512 is faster.
RSA-256 is even faster.
This question is stupid.

Context: speed

What is the fastest public-key encryption system with security level $\geq 2^b$?

Context: speed

What is the fastest public-key encryption system with security level $\geq 2^b$?

(Plausible-sounding definition: breaking costs $\geq 2^b$.)

Context: speed

What is the fastest public-key encryption system with security level $\geq 2^b$?

(Plausible-sounding definition: breaking with probability 1 costs $\geq 2^b$.)

Context: speed

What is the fastest
public-key encryption system
with security level $\geq 2^b$?

(Plausible-sounding definition:
for each $\epsilon > 0$,
breaking with probability $\geq \epsilon$
costs $\geq 2^b \epsilon$.)

Context: speed

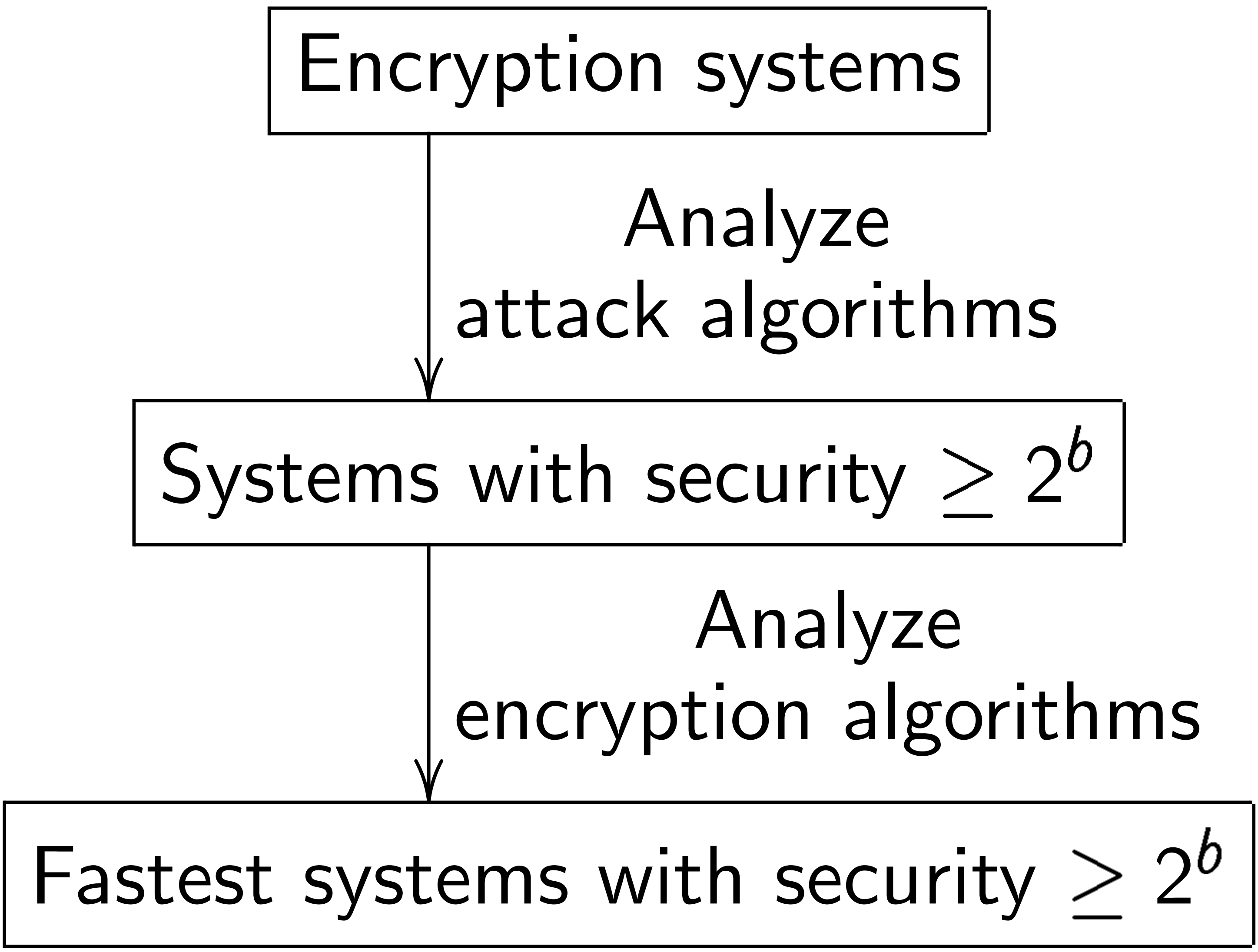
What is the fastest public-key encryption system with security level $\geq 2^b$?

(Plausible-sounding definition: for each $\epsilon > 2^{-b/2}$, breaking with probability $\geq \epsilon$ costs $\geq 2^b\epsilon$.)

Context: speed

What is the fastest public-key encryption system with security level $\geq 2^b$?

How to evaluate candidates:



Example of speed analysis

RSA (with small exponent, reasonable padding, etc.):

Factoring n costs $2^{(\lg n)^{1/3+o(1)}}$ by the number-field sieve. Conjecture: this is the optimal attack against RSA.

Key size: Can take $\lg n \in b^{3+o(1)}$ ensuring $2^{(\lg n)^{1/3+o(1)}} \geq 2^b$.

Encryption: Fast exp costs $(\lg n)^{1+o(1)}$ bit operations.

Summary: RSA costs $b^{3+o(1)}$.

ECC (with strong curve/$\mathbf{F}_q$, reasonable padding, etc.):

ECDL costs $2^{(1/2+o(1))\lg q}$ by Pollard's rho method. Conjecture: this is the optimal attack against ECC.

Can take $\lg q \in (2+o(1))b$.

Encryption: Fast scalar mult costs $(\lg q)^{2+o(1)} = b^{2+o(1)}$.

Summary: ECC costs $b^{2+o(1)}$. Asymptotically faster than RSA. Bonus: also $b^{2+o(1)}$ *decryption*.

1978 McEliece system (with length-n classical Goppa codes, reasonable padding, etc.):

Conjecture: Fastest attacks cost $2^{(\beta+o(1))n/\lg n}$.

Can take $n \in (1/\beta + o(1))b \lg b$.

Encryption: Matrix mult costs $n^{2+o(1)} = b^{2+o(1)}$.

Summary: McEliece costs $b^{2+o(1)}$.

Is this faster than ECC?
Need more detailed analysis.

ECC encryption:
$\Theta(\lg q)$ operations in $\mathbf{F}_q$.
Each operation in $\mathbf{F}_q$ costs
$\Theta(\lg q \lg\lg q \lg\lg\lg q)$.
Total $\Theta(b^2 \lg b \lg\lg b)$.

McEliece encryption,
with 1986 Niederreiter speedup:
$\Theta(n/\lg n)$ additions in $\mathbf{F}_2^n$,
each costing $\Theta(n)$.
Total $\Theta(b^2 \lg b)$.

McEliece is asymptotically faster.
Bonus: *Much* faster decryption.
Another bonus: Post-quantum.

Algorithmic advances can change this picture. Examples:

1. Speed up ECC: can reduce $\lg\lg b$ using 2007 Fürer; maybe someday eliminate $\lg\lg b$?

Algorithmic advances can change this picture. Examples:

1. Speed up ECC: can reduce $\lg \lg b$ using 2007 Fürer; maybe someday eliminate $\lg \lg b$?

2. This paper: **asymptotically faster attack on McEliece**. "Ball-collision decoding." Need larger McEliece key sizes.

Algorithmic advances can change this picture. Examples:

1. Speed up ECC: can reduce $\lg\lg b$ using 2007 Fürer; maybe someday eliminate $\lg\lg b$?

2. This paper: **asymptotically faster attack on McEliece**. "Ball-collision decoding." Need larger McEliece key sizes.

3. Ongoing: we're optimizing "subfield AG" variant of McEliece. Conjecture: Fastest attacks cost $2^{(\alpha+o(1))n}$; encryption costs $\Theta(b^2)$.

Generic decoding algorithms

Some history: 1962 Prange; 1981 Clark (crediting Omura); 1988 Lee–Brickell; 1988 Leon; 1989 Krouk; 1989 Stern; 1989 Dumer; 1990 Coffey–Goodman; 1990 van Tilburg; 1991 Dumer; 1991 Coffey–Goodman–Farrell; 1993 Chabanne–Courteau; 1993 Chabaud; 1994 van Tilburg; 1994 Canteaut–Chabanne; 1998 Canteaut–Chabaud; 1998 Canteaut–Sendrier; 2008 B.–L.–P.: 2009 Finiasz–Sendrier; 2010 P.; 2011 B.–L.–P, this paper.

A typical decoding problem

Input: 500-bit vector s; and a 900×500 matrix of bits.

Goal: Find 50 rows with xor s.

...11001...	r_1
...10111...	r_2
...10101...	r_3
$\vdots$	$\vdots$
...01011...	r_{900}
...01010...	s

A typical decoding problem

Input: 500-bit vector s; and a 900×500 matrix of bits.

Goal: Find 50 rows with xor s.

$\ldots 11001 \ldots$	r_1
$\ldots 10111 \ldots$	r_2
$\ldots 10101 \ldots$	r_3
$\vdots$	$\vdots$
$\ldots 01011 \ldots$	r_{900}
$\ldots 01010 \ldots$	$s = r_2 \oplus r_7 \oplus r_{34} \oplus r$

Row randomization

Can arbitrarily permute rows without changing problem.

Goal: Find 50 rows with xor s.

. . . 11001 . . .	r_1
. . . 10111 . . .	r_2
. . . 10101 . . .	r_3
$\vdots$	$\vdots$
. . . 01011 . . .	r_{900}
. . . 01010 . . .	$s = r_2 \oplus r_7 \oplus r_{34} \oplus r$

Row randomization

Can arbitrarily permute rows without changing problem.

Goal: Find 50 rows with xor s.

...10111...	r_1
...11001...	r_2
...10101...	r_3
⋮	⋮
...01011...	r_{900}
...01010...	$s = r_1 \oplus r_7 \oplus r_{34} \oplus r$

Column normalization

Can also permute columns
without changing problem.
Goal: Find 50 rows with xor s.

$\ldots 10111 \ldots$	r_1
$\ldots 11001 \ldots$	r_2
$\ldots 10101 \ldots$	r_3
$\vdots$	$\vdots$
$\ldots 01011 \ldots$	r_{900}
$\ldots 01010 \ldots$	$s = r_1 \oplus r_7 \oplus r_{34} \oplus r$

Column normalization

Can also permute columns without changing problem.

Goal: Find 50 rows with xor s.

. . . 01111 . . .	r_1
. . . 11001 . . .	r_2
. . . 01101 . . .	r_3
⋮	⋮
. . . 10011 . . .	r_{900}
. . . 10010 . . .	$s = r_1 \oplus r_7 \oplus r_{34} \oplus r$

Systematic form

Can add one column to another.

$\Rightarrow$ Build an identity matrix.

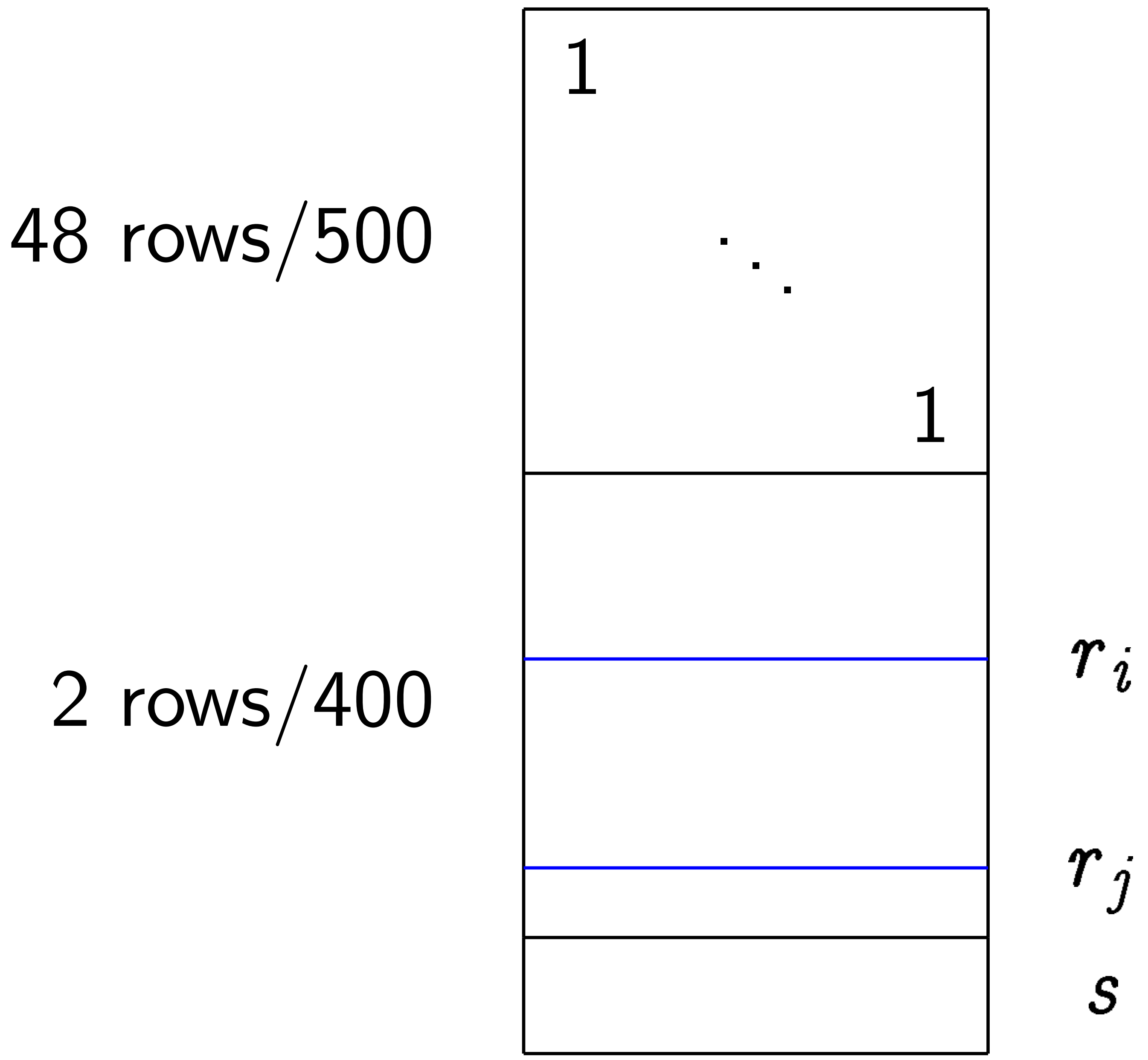
Goal: Find 50 rows with xor s.

1000 . . . 0000	r_1
0100 . . . 0000	r_2
0010 . . . 0000	r_3
$\ddots$	$\vdots$
0000 . . . 0001	r_{500}
1010 . . . 1100	r_{501}
$\vdots$	$\vdots$
1101 . . . 0111	r_{900}
0110 . . . 0000	$s = r_2 \oplus r_3 \oplus r_{18} \oplus r$

1962 Prange, basic
information-set decoding:
Maybe xor involves
none of last 400 rows.
If so, immediately see that
s has weight 50. Done!
If not, re-randomize and restart.

1962 Prange, basic
information-set decoding:
Maybe xor involves
none of last 400 rows.
If so, immediately see that
s has weight 50. Done!
If not, re-randomize and restart.

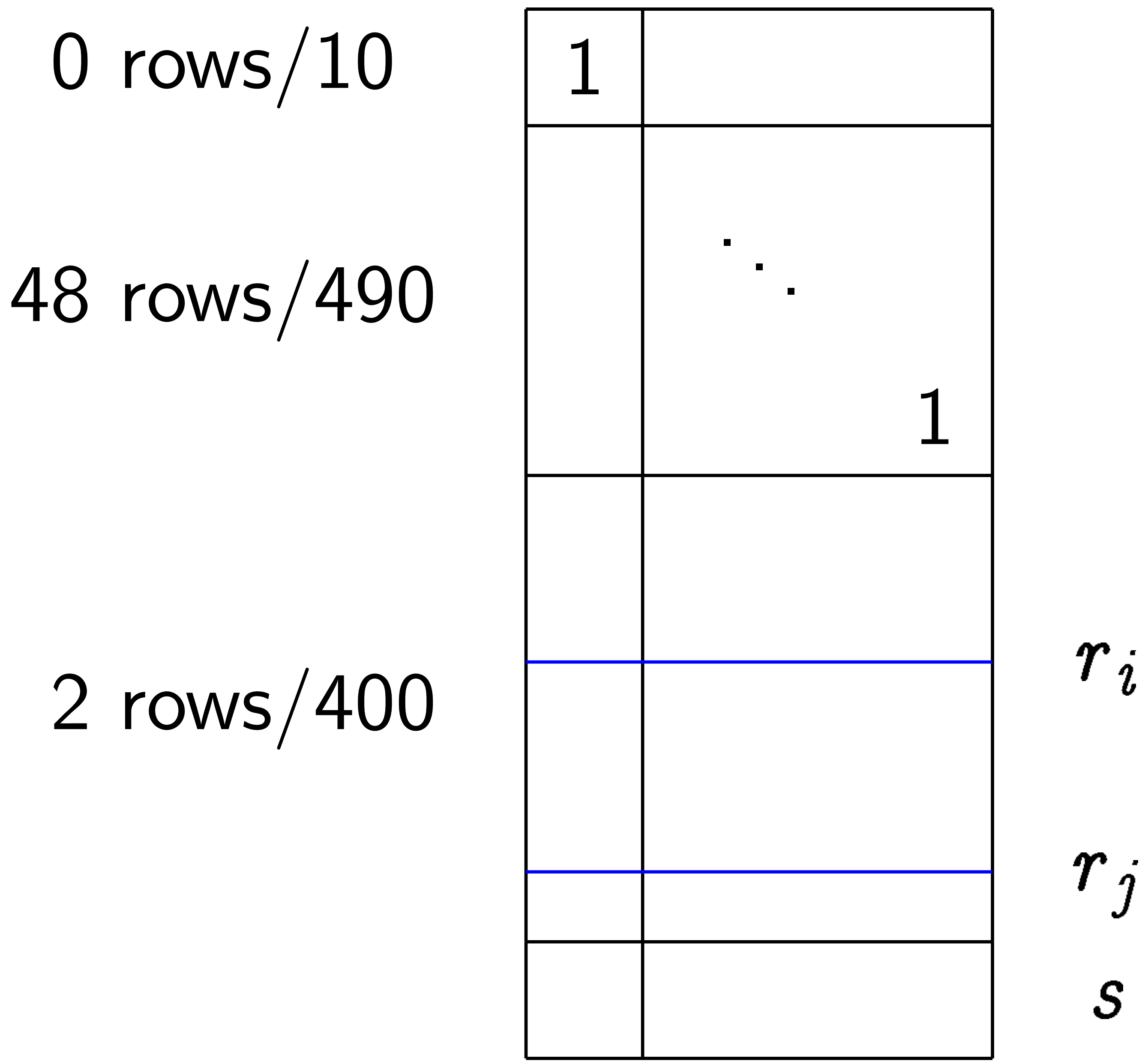
1988 Lee–Brickell:
More likely that xor involves
exactly 2 of last 400 rows.
Check for each i, j whether
$s \oplus r_i \oplus r_j$ has weight 48.

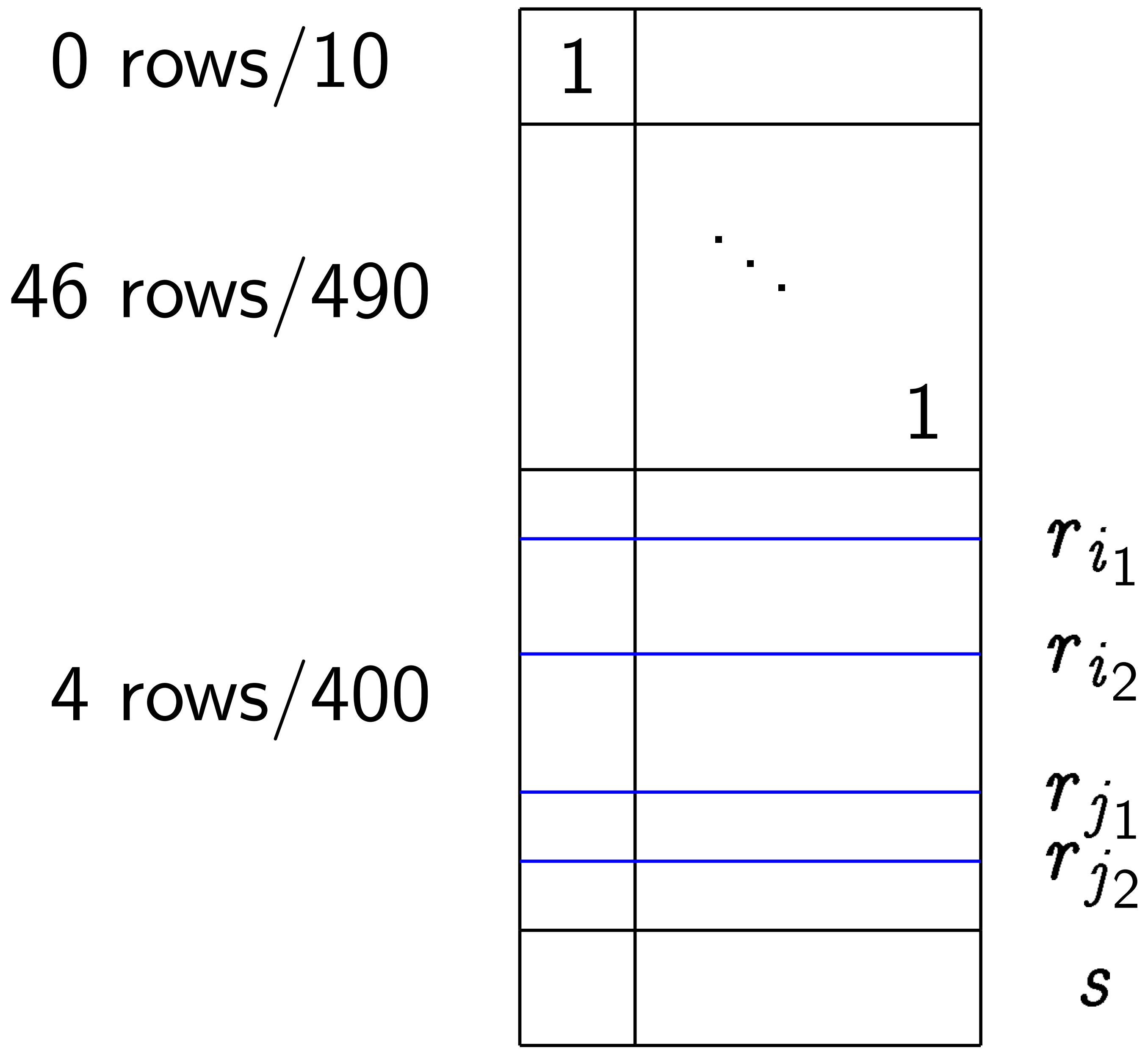

48 rows/500
1
1
2 rows/400
r_i
r_j
s

1989 Leon, 1989 Krouk:
Check for each i, j whether
$s \oplus r_i \oplus r_j$ has weight 48
with first 10 bits all zero.
Much faster to test,
not much loss in success chance.

1989 Leon, 1989 Krouk:
Check for each i, j whether
$s \oplus r_i \oplus r_j$ has weight 48
with first 10 bits all zero.
Much faster to test,
not much loss in success chance.

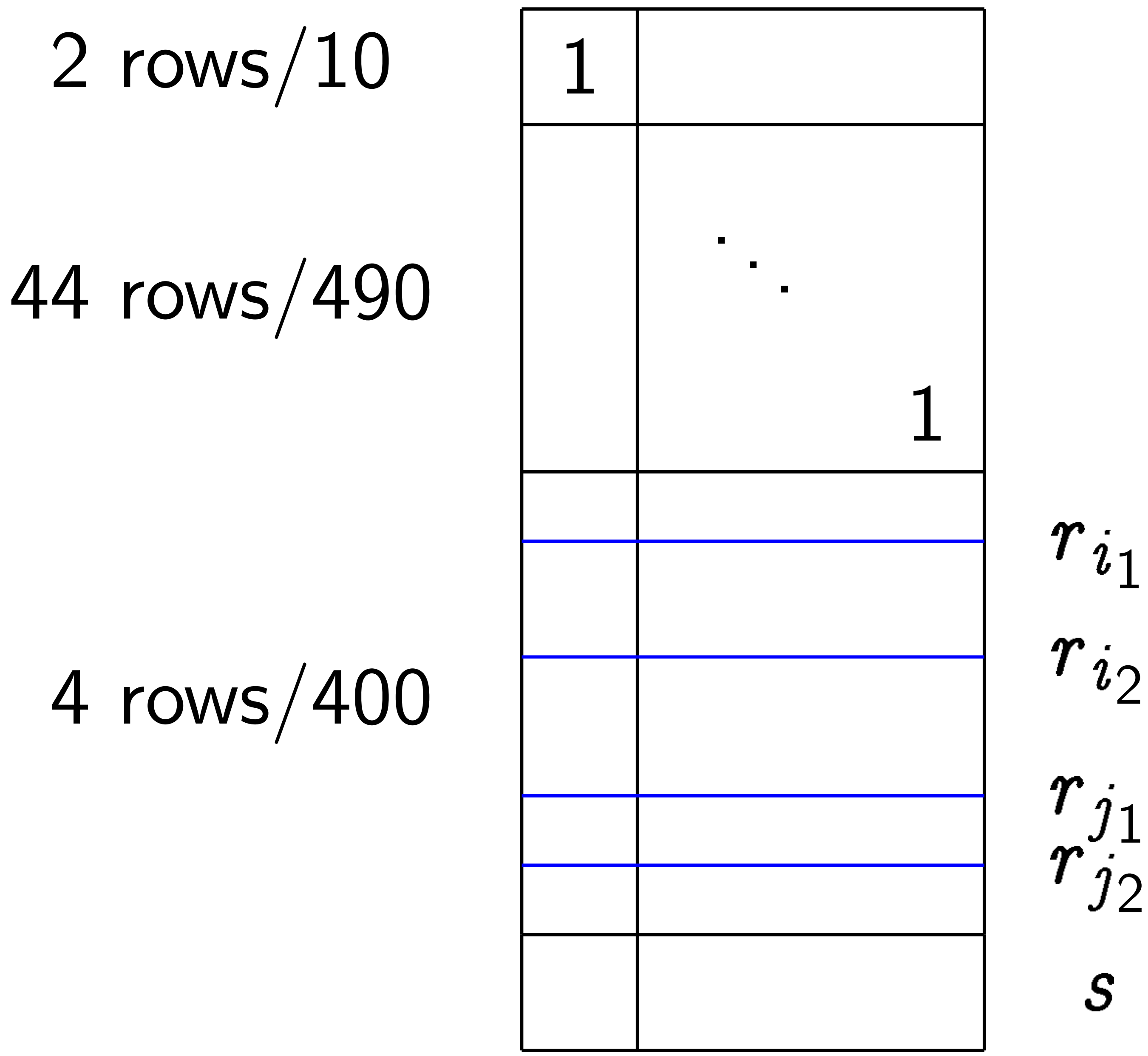
1989 Stern, **collision decoding**:
$\sqrt{\quad}$ speedup!
Find collisions between
first 10 bits of $s \oplus r_i$
and first 10 bits of r_j.
For each collision, check whether
$s \oplus r_i \oplus r_j$ has weight 48.


0 rows/10
48 rows/490
2 rows/400
1
1
r_i
r_j
s



Or $s \oplus r_{i_1} \oplus \cdots \oplus r_{i_p}$
and $r_{j_1} \oplus \cdots \oplus r_{j_p}$.
Optimize choice of p.
Of course, also optimize 10 etc.

New, **ball-collision decoding**:
Find collisions between (e.g.) weight-1 Hamming ball around first 10 bits of $s \oplus r_{i_1} \oplus r_{i_2}$ and weight-1 Hamming ball around first 10 bits of $r_{j_1} \oplus r_{j_2}$.



Our main theorem:

For w rows of $n \times (n-k)$ matrix, constant $w/n, k/n$ as $n \to \infty$, under standard assumptions, optimized collision decoding costs $2^{(\alpha+o(1))n}$ and optimized ball-collision decoding costs $2^{(\alpha'+o(1))n}$ with $\alpha' < \alpha$.

See cr.yp.to/ballcoll.html:

- proof of smaller exponents;
- conservative lower bounds;
- complete reference software.