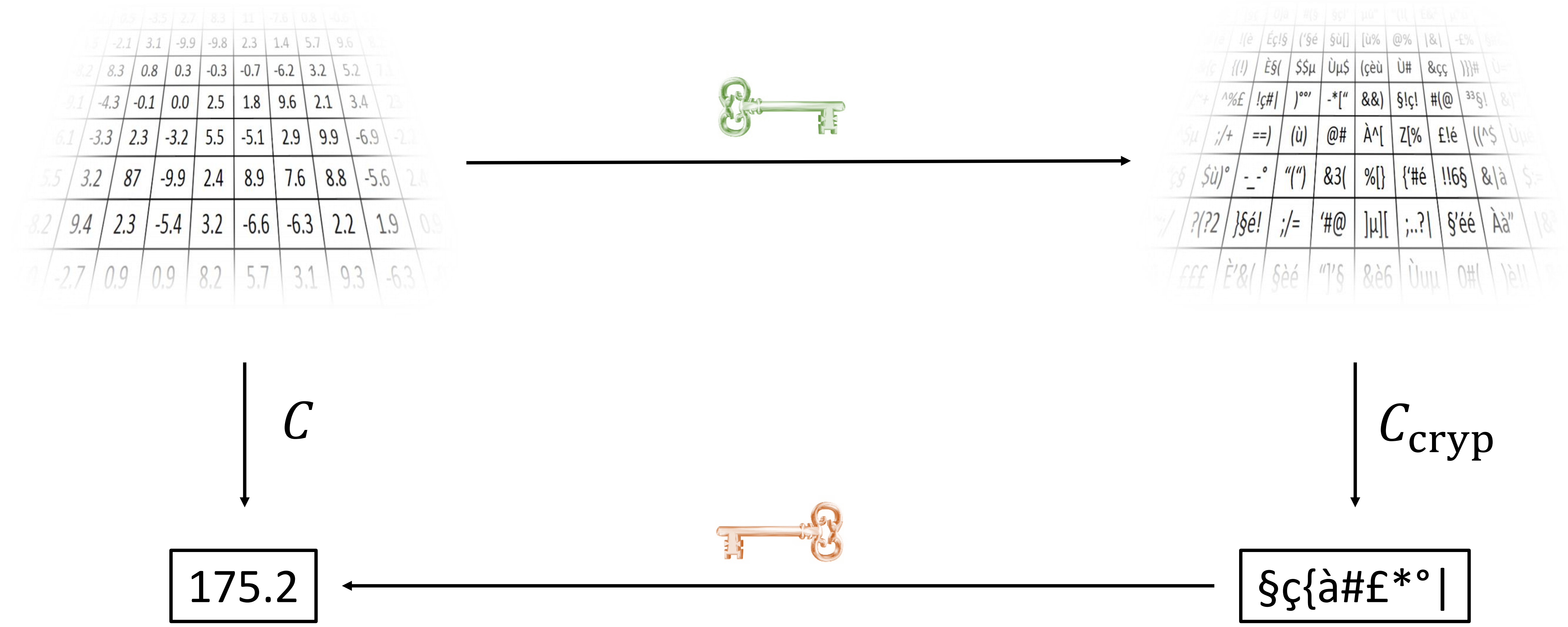
Homomorphic SIM^{2}D operations: Single Instruction Much More Data

Wouter Castryck
Ilia Iliashenko
Frederik Vercauteren

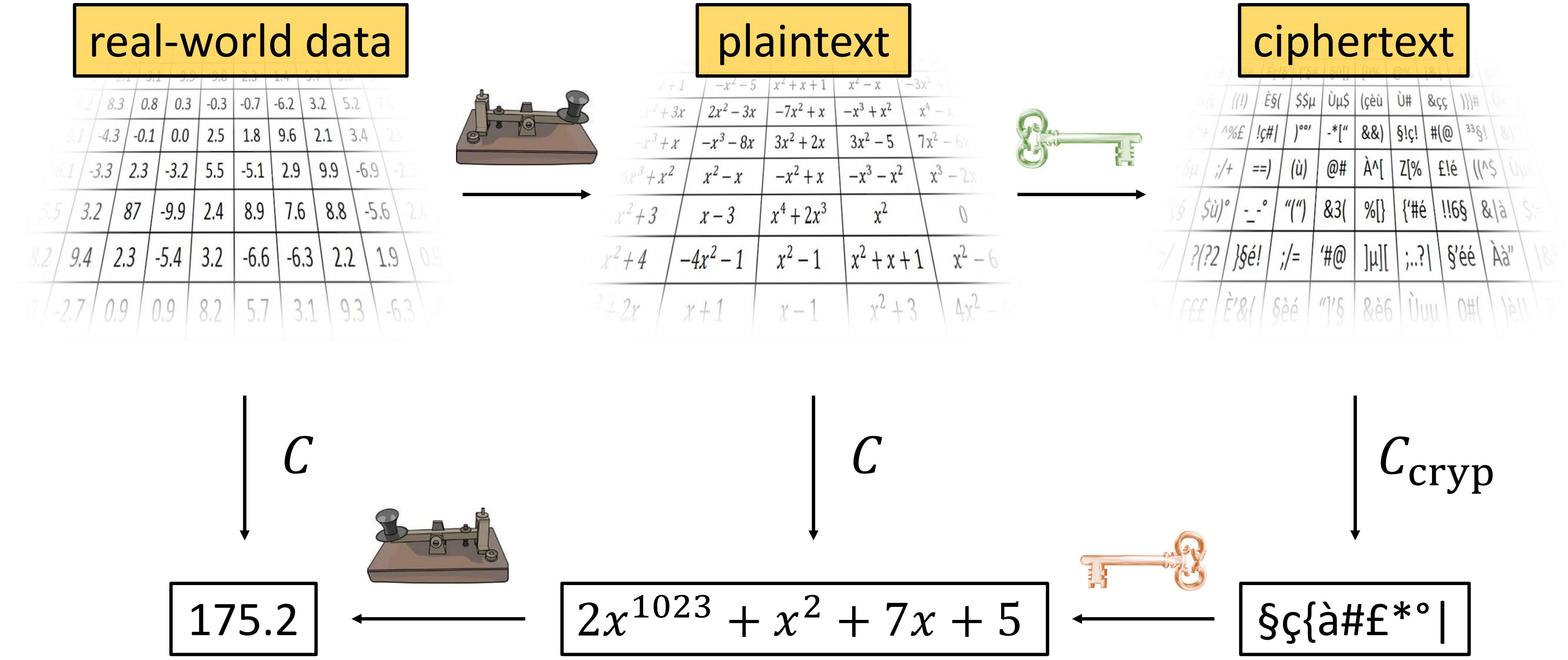
KU LEUVEN
imec
COSIC

P M '29

Homomorphic encryption



Homomorphic encoding

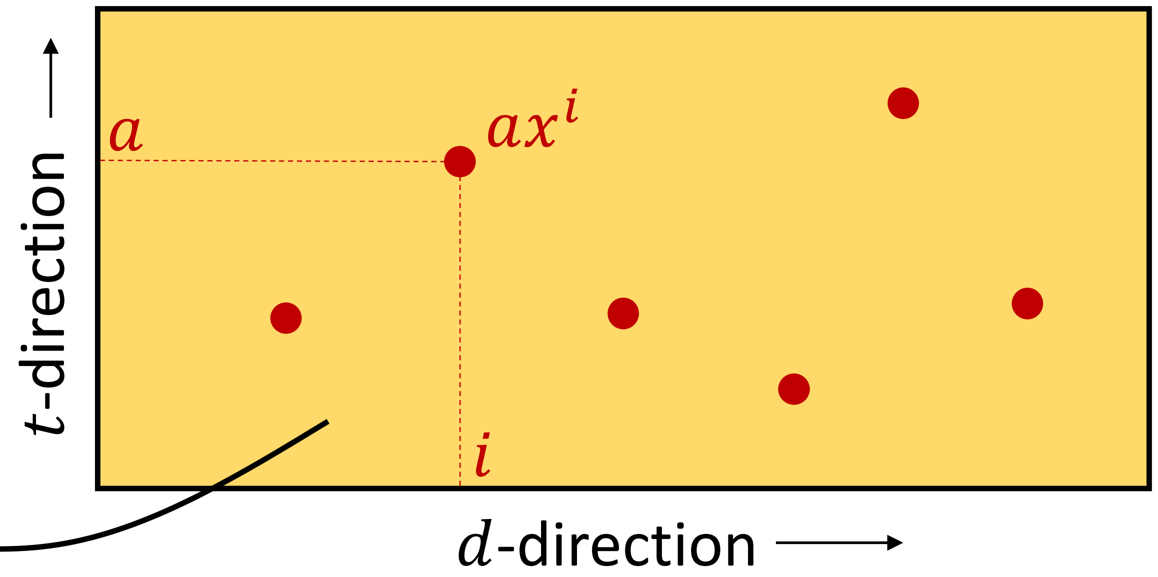


Plaintext space

Typically a ring of the form $R_t = \dfrac{\mathbf{Z}[x]}{(f(x), t)}$

where $t \in \mathbf{Z}_{\geq 2}$ and $f(x) \in \mathbf{Z}[x]$ is monic irreducible of degree d.

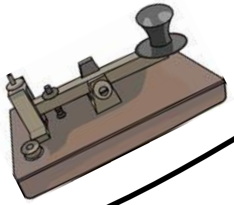
We represent this by a box:

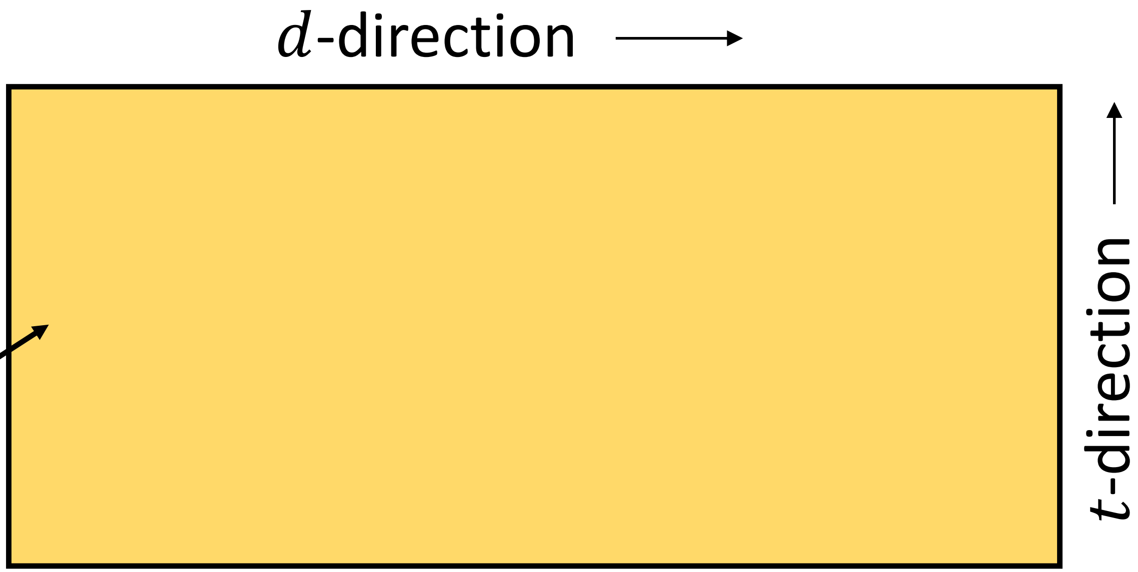


Polynomials of degree $< d$
and coefficients in $[0, t)$.

Homomorphic encoding

How to encode
real-world input θ?





General principle: find an integer-digit expansion

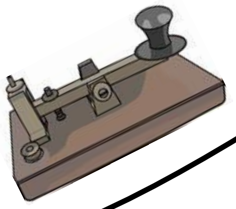
$$\theta \approx a_r b^r + a_{r-1} b^{r-1} + \cdots + a_1 b + a_0 \quad \text{for some base } b \in \mathbf{C}.$$

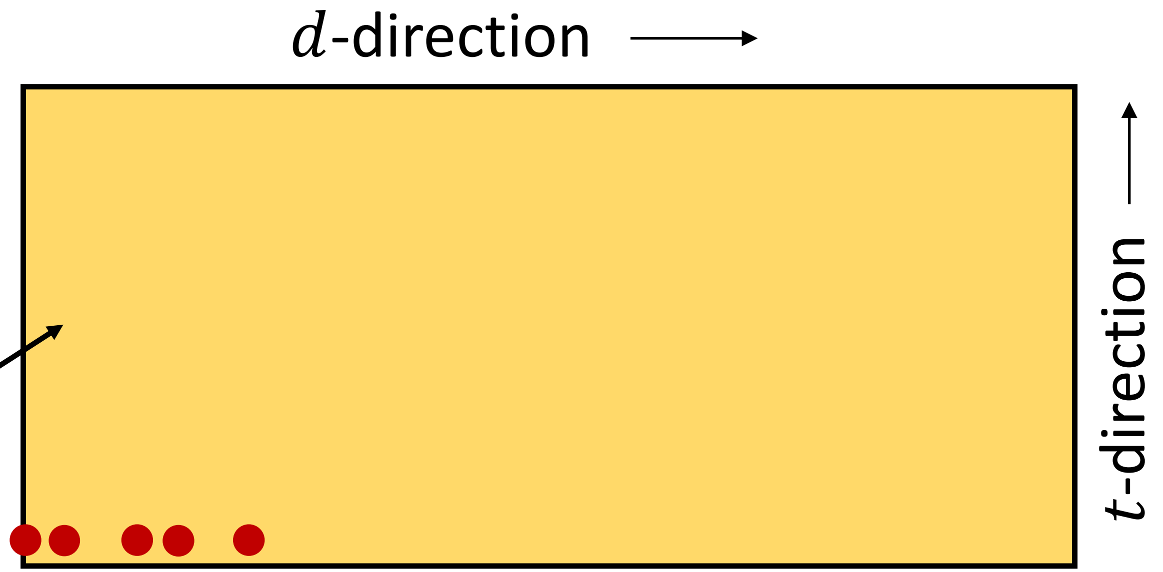
Then encode as $a_r x^r + a_{r-1} x^{r-1} + \cdots + a_1 x + a_0$.

Decoding: evaluate in $x = b$. Works well if no *overflow*.

Homomorphic encoding

How to encode
real-world input θ?





General principle: find an integer-digit expansion

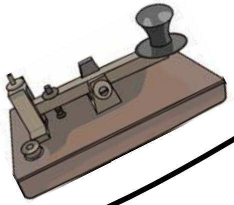
$$\boldsymbol{\theta = 2^6 + 2^4 + 2^3 + 2 + 1}$$

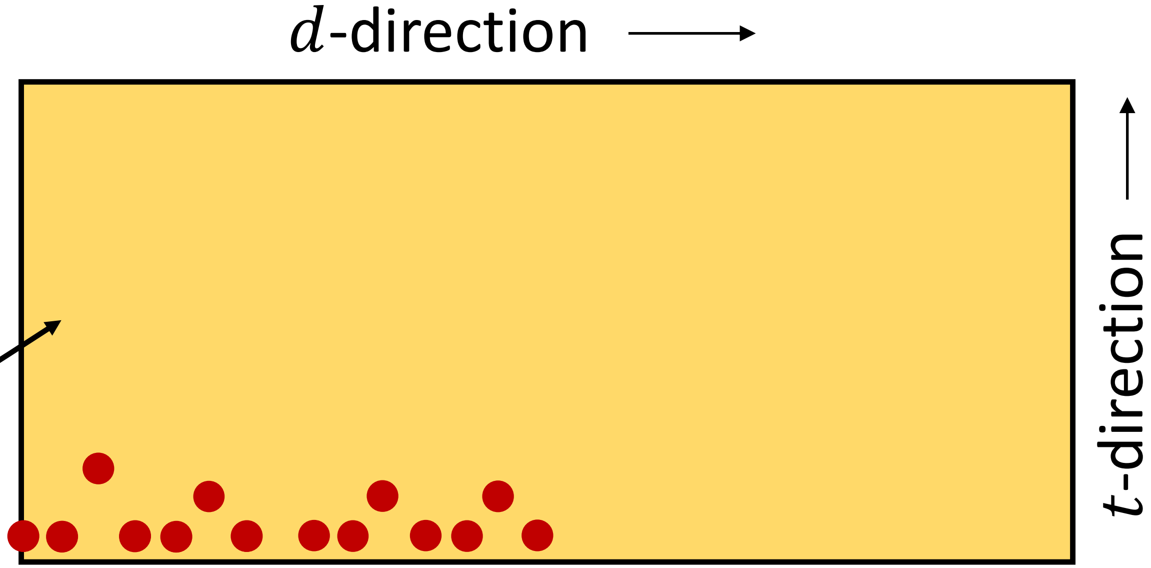
Then encode as $a_r x^r + a_{r-1} x^{r-1} + \cdots + a_1 x + a_0$.

Decoding: evaluate in $x = b$. Works well if no *overflow*.

Homomorphic encoding

How to encode
real-world input θ?





General principle: find an integer-digit expansion

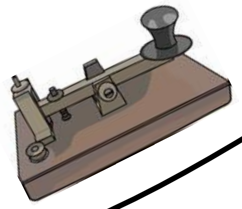
$$\boldsymbol{\theta = 2^6 + 2^4 + 2^3 + 2 + 1}$$

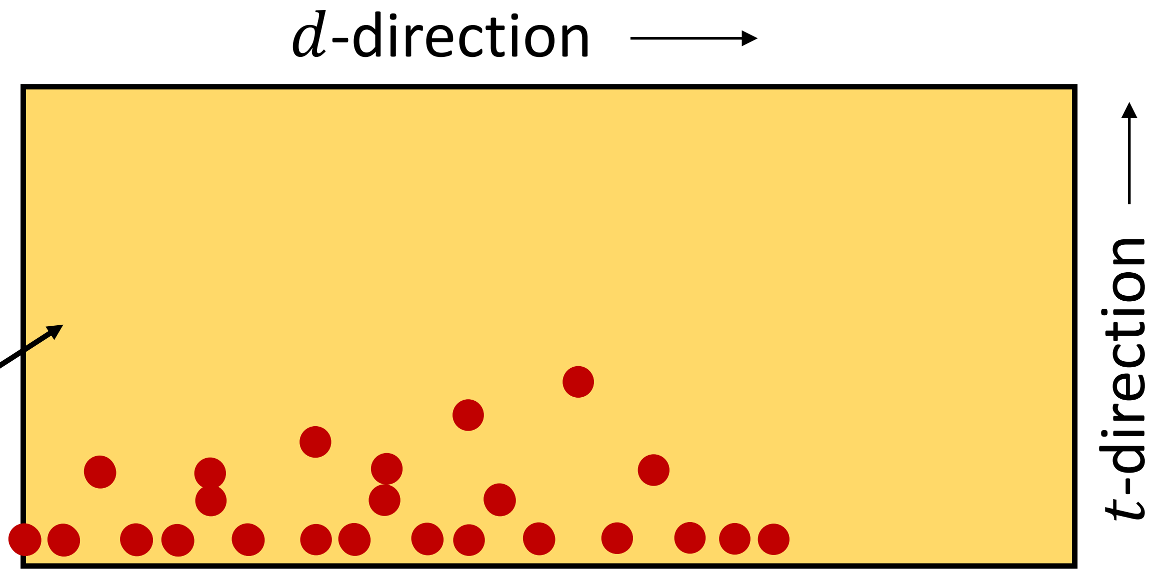
Then encode as $a_r x^r + a_{r-1} x^{r-1} + \cdots + a_1 x + a_0$.

Decoding: evaluate in $x = b$. Works well if no *overflow*.

Homomorphic encoding

How to encode
real-world input θ?





General principle: find an integer-digit expansion

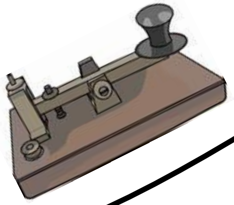
$$\boldsymbol{\theta = 2^6 + 2^4 + 2^3 + 2 + 1}$$

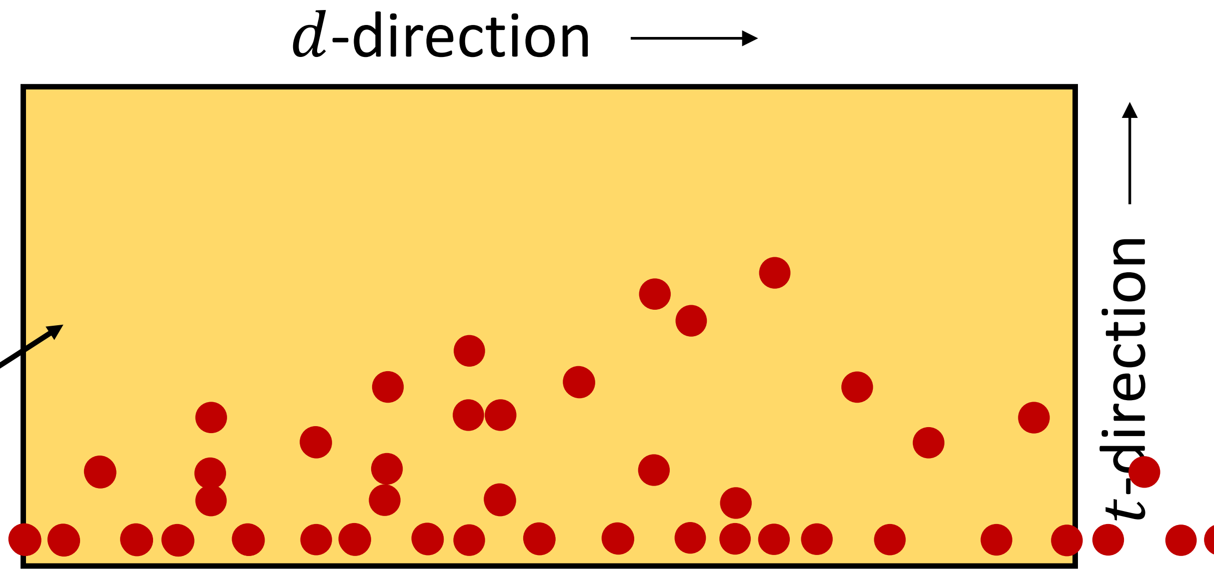
Then encode as $a_r x^r + a_{r-1} x^{r-1} + \cdots + a_1 x + a_0$.

Decoding: evaluate in $x = b$. Works well if no *overflow*.

Homomorphic encoding

How to encode
real-world input θ?





General principle: find an integer-digit expansion

$$\boldsymbol{\theta = 2^6 + 2^4 + 2^3 + 2 + 1}$$

Then encode as $a_r x^r + a_{r-1} x^{r-1} + \cdots + a_1 x + a_0$.

Decoding: evaluate in $x = b$. Works well if no *overflow*.

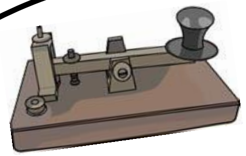
Fractional encoding

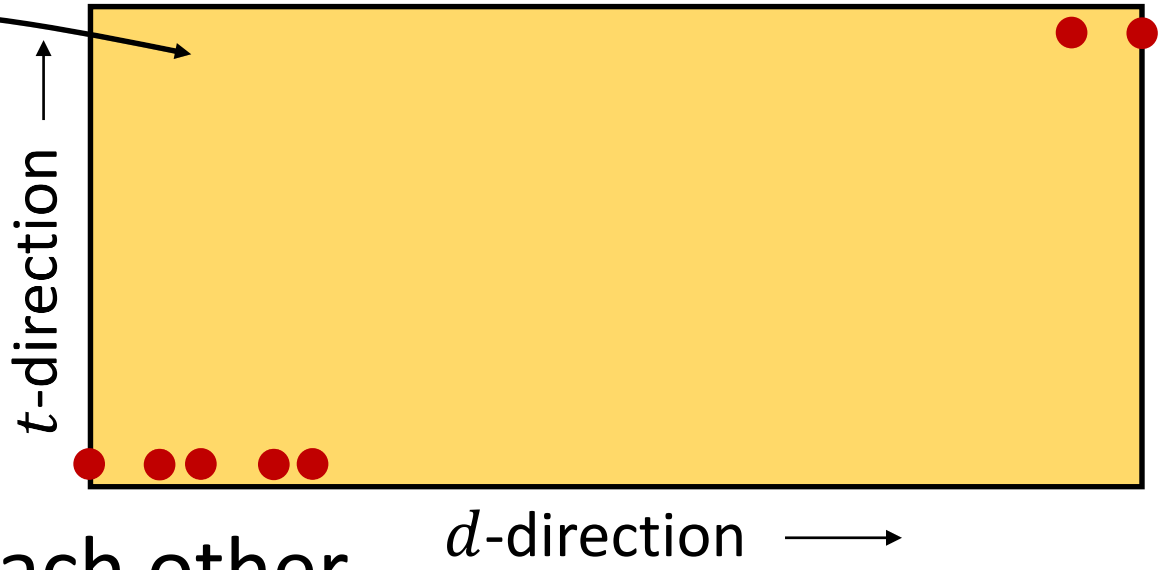
Encoding fractional expansions

$$\theta \approx a_r b^r + \cdots + a_1 b + a_0 + a_{-1} b^{-1} + \cdots + a_{-s} b^{-s}?$$

[Dowlin et al., '15] If $f(x) = x^d + 1$ then $x^{-i} \equiv -x^{d-i}$, so: put fractional part at the high powers, with negated sign.

$$\boldsymbol{\theta = 2^6 + 2^4 + 2^3 + 2 + 1 + 2^{-1} + 2^{-3}}$$





Works as long as high powers and low powers do not *overflow* each other.

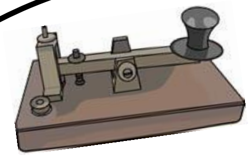
Fractional encoding

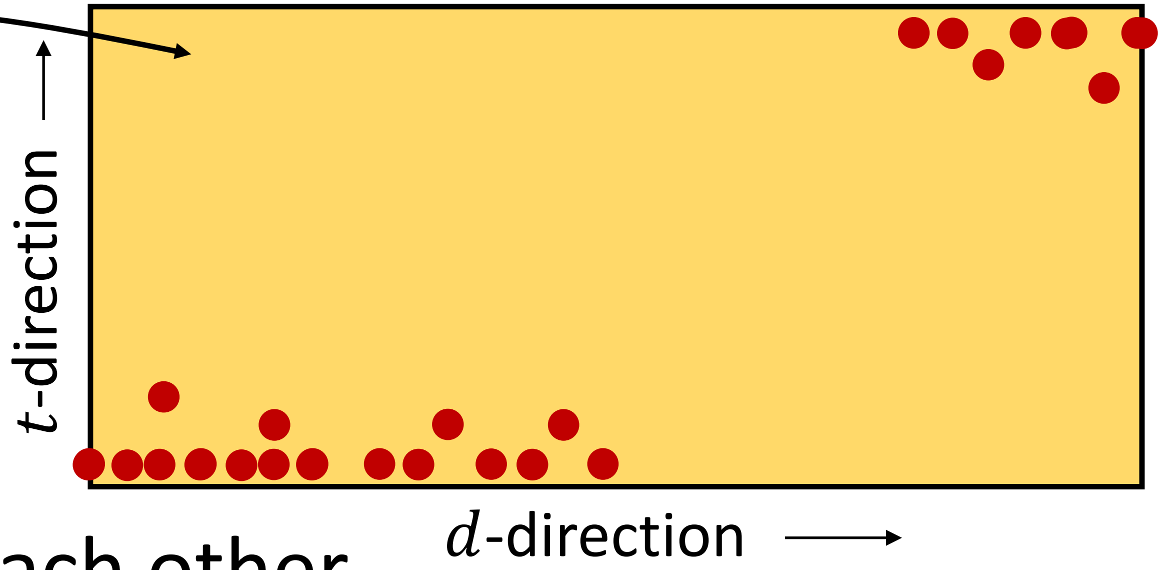
Encoding fractional expansions

$$\theta \approx a_r b^r + \cdots + a_1 b + a_0 + a_{-1} b^{-1} + \cdots + a_{-s} b^{-s}?$$

[Dowlin et al., '15] If $f(x) = x^d + 1$ then $x^{-i} \equiv -x^{d-i}$, so: put fractional part at the high powers, with negated sign.

$$\boldsymbol{\theta = 2^6 + 2^4 + 2^3 + 2 + 1 + 2^{-1} + 2^{-3}}$$





Works as long as high powers and low powers do not *overflow* each other.

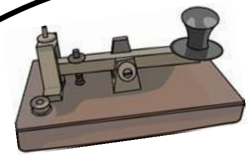
Fractional encoding

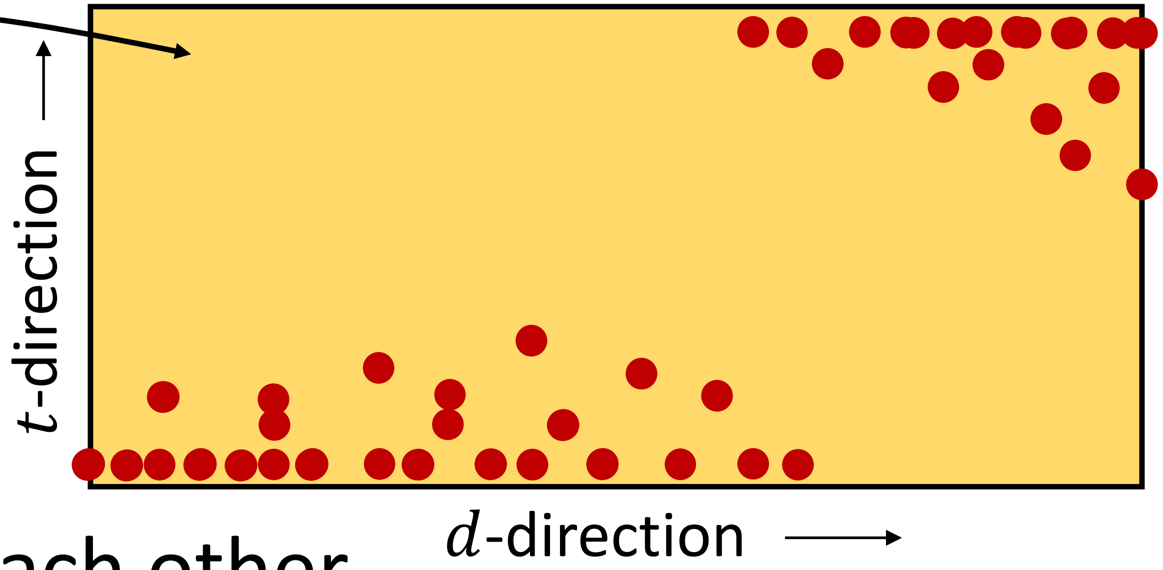
Encoding fractional expansions

$$\theta \approx a_r b^r + \cdots + a_1 b + a_0 + a_{-1} b^{-1} + \cdots + a_{-s} b^{-s}?$$

[Dowlin et al., '15] If $f(x) = x^d + 1$ then $x^{-i} \equiv -x^{d-i}$, so: put fractional part at the high powers, with negated sign.

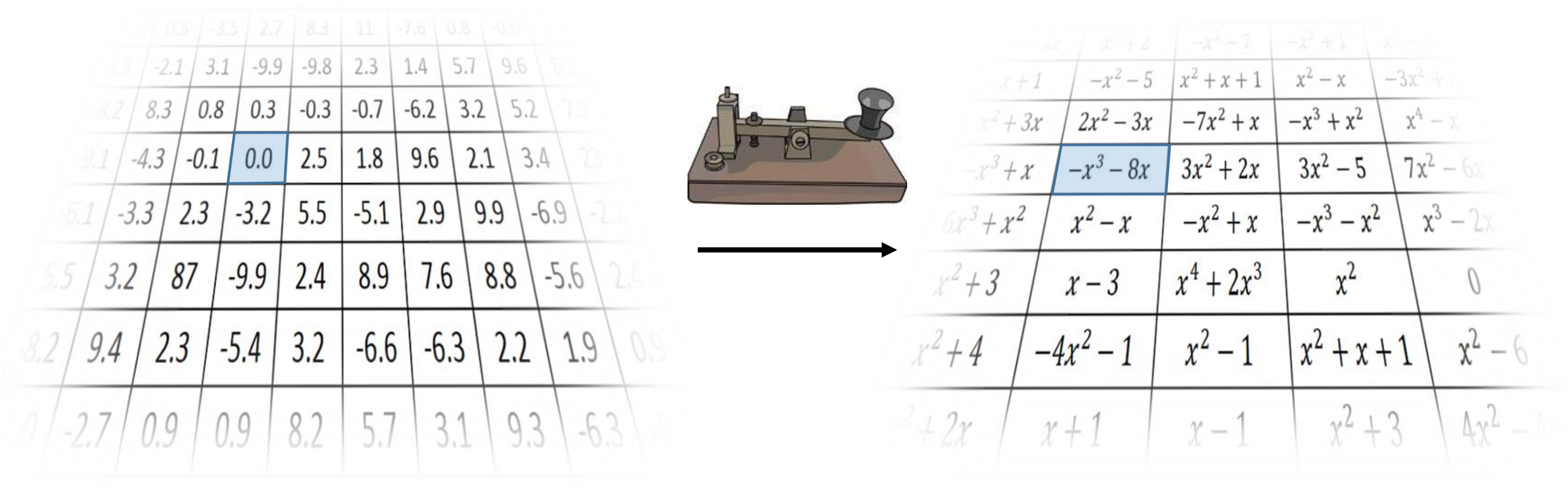
$$\boldsymbol{\theta = 2^6 + 2^4 + 2^3 + 2 + 1 + 2^{-1} + 2^{-3}}$$





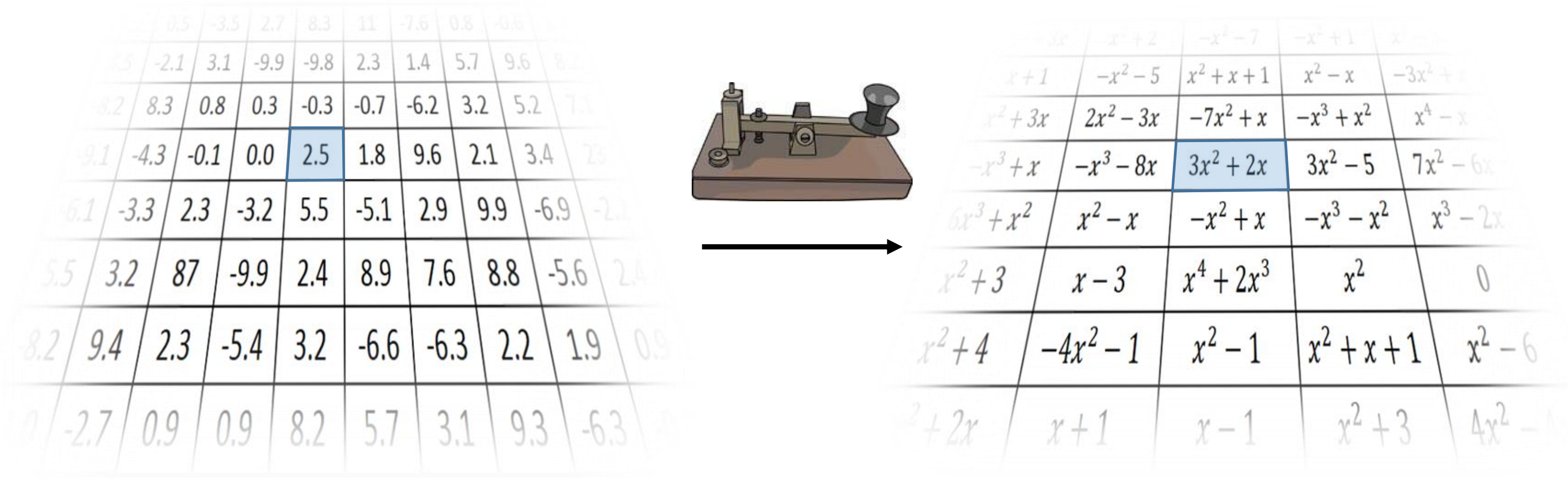
Works as long as high powers and low powers do not *overflow* each other.

SIMD



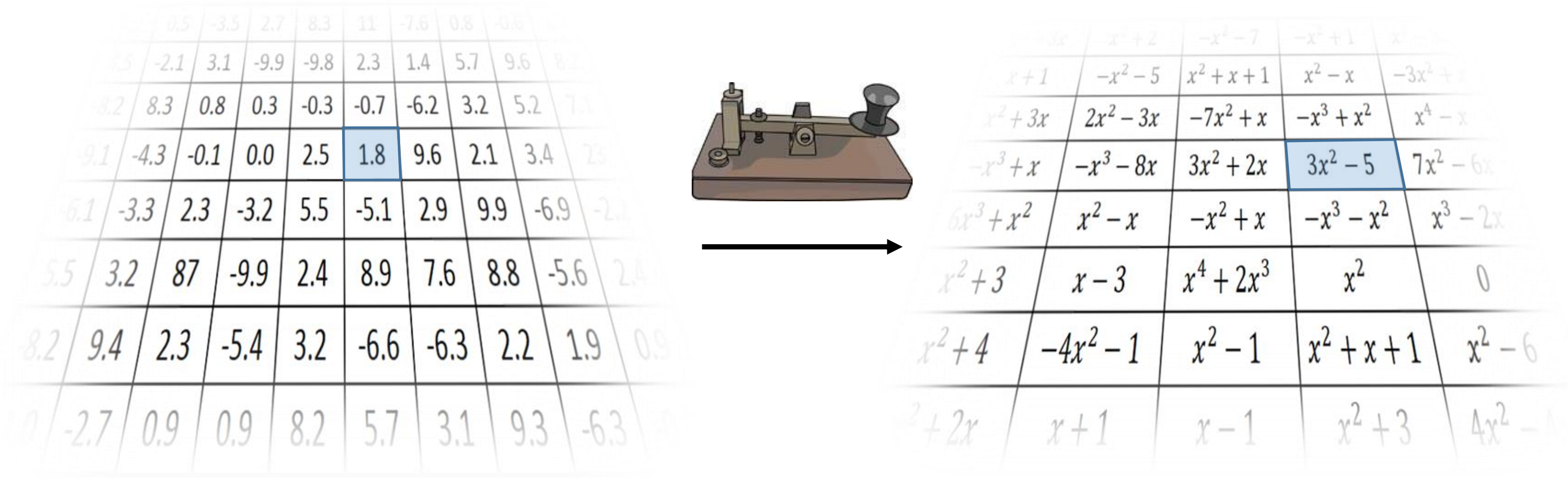
8.3 0.8 0.3 -0.3 -0.7 -6.2 3.2 5.2
-4.3 -0.1 0.0 2.5 1.8 9.6 2.1 3.4
-3.3 2.3 -3.2 5.5 -5.1 2.9 9.9 -6.9
3.2 87 -9.9 2.4 8.9 7.6 8.8 -5.6
9.4 2.3 -5.4 3.2 -6.6 -6.3 2.2 1.9
-2.7 0.9 0.9 8.2 5.7 3.1 9.3 -6.3
$-x^2-5$ x^2+x+1 x^2-x
$2x^2-3x$ $-7x^2+x$ $-x^3+x^2$
$-x^3-8x$ $3x^2+2x$ $3x^2-5$
x^2-x $-x^2+x$ $-x^3-x^2$
$x-3$ x^4+2x^3 x^2 0
$-4x^2-1$ x^2-1 x^2+x+1
$x+1$ $x-1$ x^2+3

SIMD



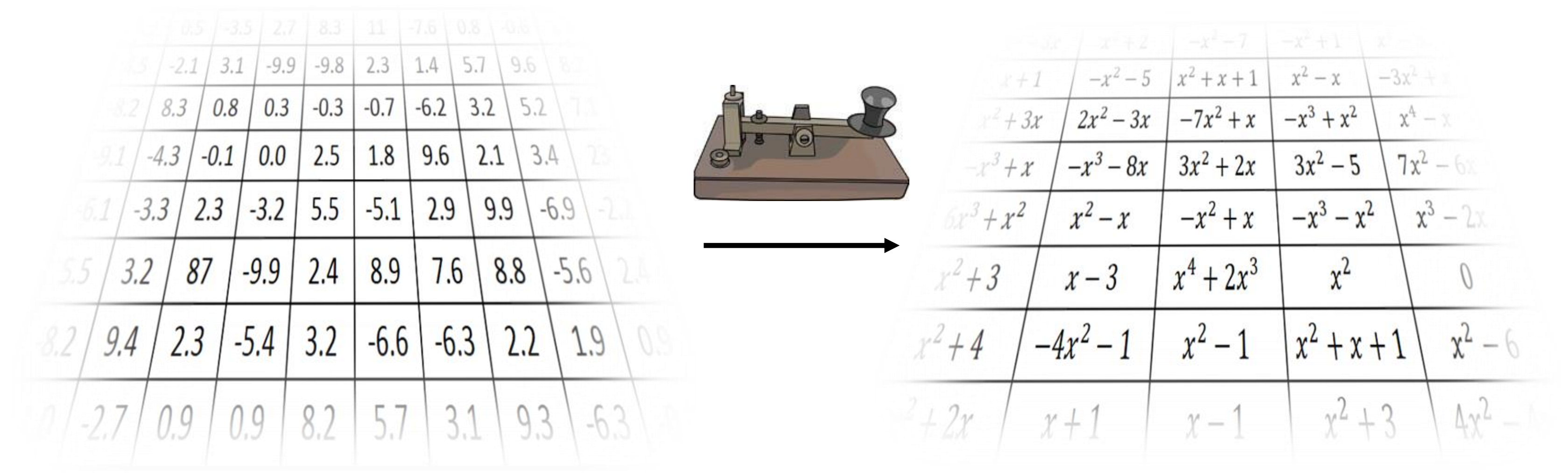
-2.1 3.1 -9.9 -9.8 2.3 1.4 5.7 9.6
8.3 0.8 0.3 -0.3 -0.7 -6.2 3.2 5.2
-4.3 -0.1 0.0 2.5 1.8 9.6 2.1 3.4
-3.3 2.3 -3.2 5.5 -5.1 2.9 9.9 -6.9
3.2 87 -9.9 2.4 8.9 7.6 8.8 -5.6
9.4 2.3 -5.4 3.2 -6.6 -6.3 2.2 1.9
-2.7 0.9 0.9 8.2 5.7 3.1 9.3 -6.3
$-x^2-5$ x^2+x+1 x^2-x
$2x^2-3x$ $-7x^2+x$ $-x^3+x^2$
$-x^3-8x$ $3x^2+2x$ $3x^2-5$
x^2-x $-x^2+x$ $-x^3-x^2$
$x-3$ x^4+2x^3 x^2 0
$-4x^2-1$ x^2-1 x^2+x+1
$x+1$ $x-1$ x^2+3

SIMD



-2.1 3.1 -9.9 -9.8 2.3 1.4 5.7 9.6
8.3 0.8 0.3 -0.3 -0.7 -6.2 3.2 5.2
-4.3 -0.1 0.0 2.5 1.8 9.6 2.1 3.4
-3.3 2.3 -3.2 5.5 -5.1 2.9 9.9 -6.9
3.2 87 -9.9 2.4 8.9 7.6 8.8 -5.6
9.4 2.3 -5.4 3.2 -6.6 -6.3 2.2 1.9
-2.7 0.9 0.9 8.2 5.7 3.1 9.3 -6.3
$-x^2-5$ x^2+x+1 x^2-x
$2x^2-3x$ $-7x^2+x$ $-x^3+x^2$
$-x^3-8x$ $3x^2+2x$ $3x^2-5$
x^2-x $-x^2+x$ $-x^3-x^2$
$x-3$ x^4+2x^3 x^2 0
$-4x^2-1$ x^2-1 x^2+x+1
$x+1$ $x-1$ x^2+3

SIMD

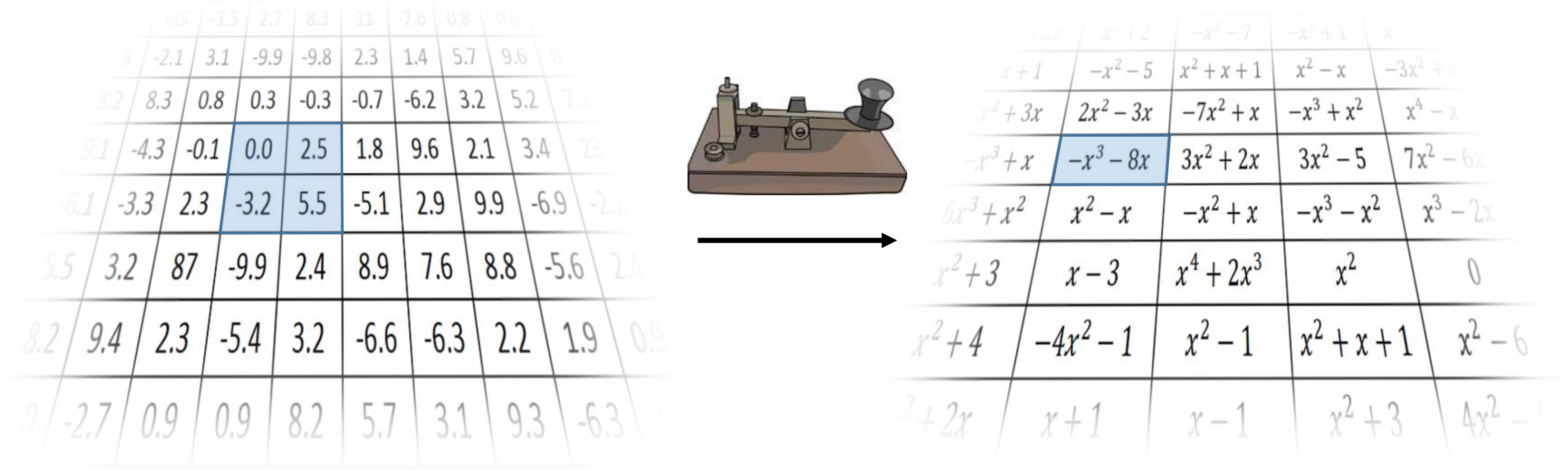


Batch encoding is possible thanks to CRT [Smart-Vercauteren, '14]:

$$R_t = \frac{\mathbf{Z}[x]}{(f(x), t)} \xrightarrow{\cong} \frac{\mathbf{Z}[x]}{(f_1(x), t)} \times \frac{\mathbf{Z}[x]}{(f_2(x), t)} \times \cdots \times \frac{\mathbf{Z}[x]}{(f_r(x), t)}$$

where the $f_i(x)$ are coprime factors of $f(x)$ modulo t.

SIMD

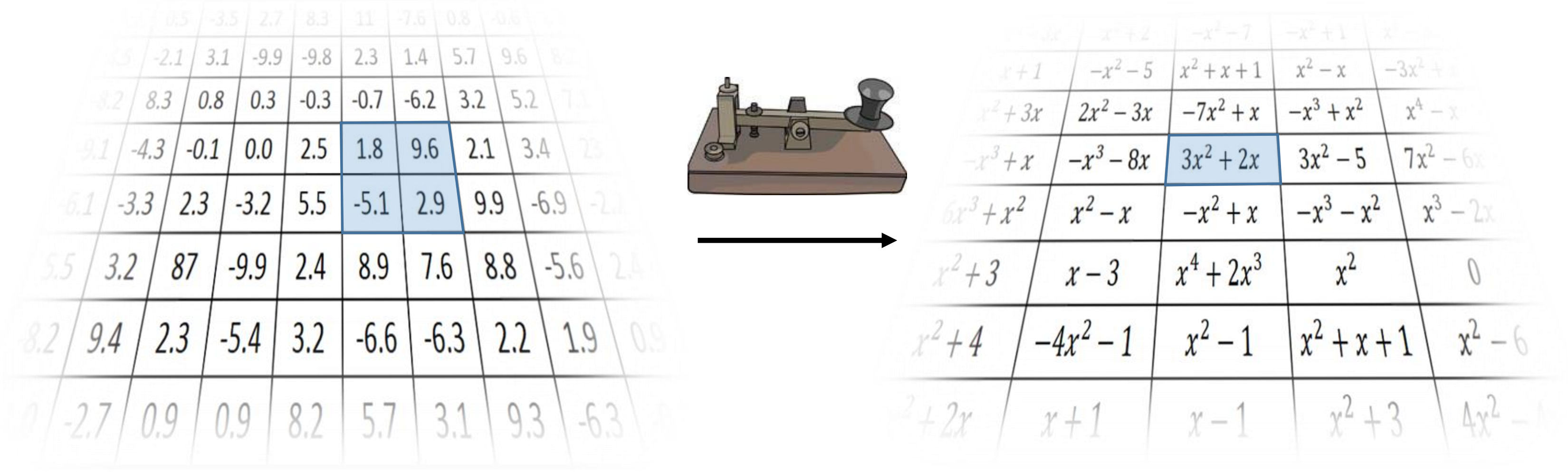


Batch encoding is possible thanks to CRT [Smart-Vercauteren, ‘14]:

$$R_t = \frac{\mathbf{Z}[x]}{(f(x),t)} \xrightarrow{\cong} \frac{\mathbf{Z}[x]}{(f_1(x),t)} \times \frac{\mathbf{Z}[x]}{(f_2(x),t)} \times \cdots \times \frac{\mathbf{Z}[x]}{(f_r(x),t)}$$

where the $f_i(x)$ are coprime factors of $f(x)$ modulo t.

SIMD

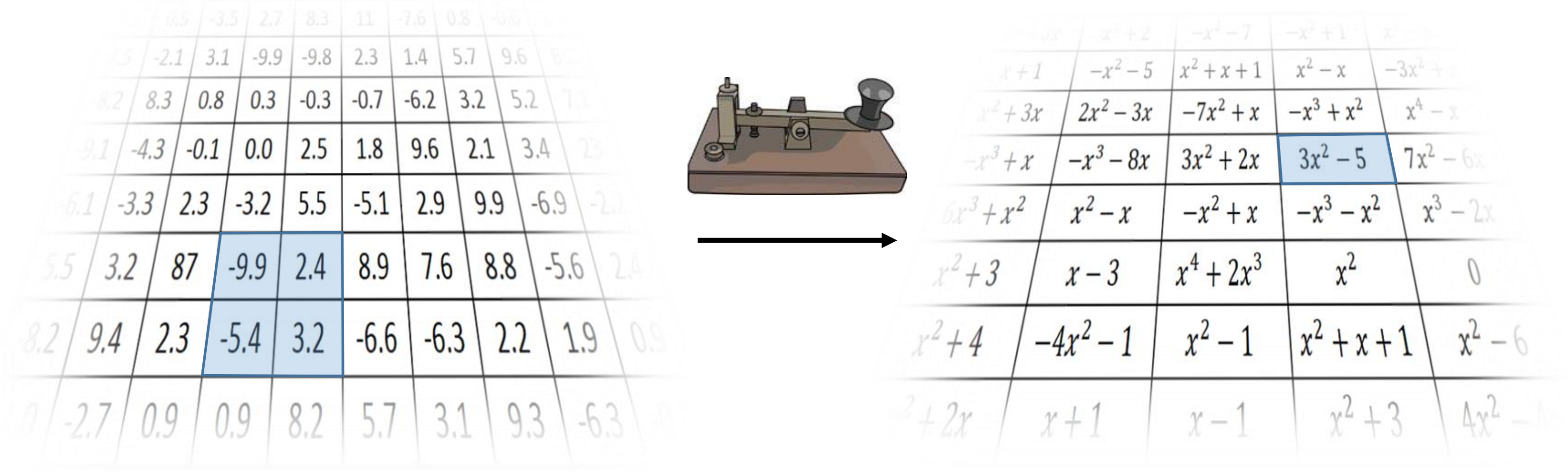


Batch encoding is possible thanks to CRT [Smart-Vercauteren, ‘14]:

$$R_t = \frac{\mathbf{Z}[x]}{(f(x),t)} \xrightarrow{\cong} \frac{\mathbf{Z}[x]}{(f_1(x),t)} \times \frac{\mathbf{Z}[x]}{(f_2(x),t)} \times \cdots \times \frac{\mathbf{Z}[x]}{(f_r(x),t)}$$

where the $f_i(x)$ are coprime factors of $f(x)$ modulo t.

SIMD



Batch encoding is possible thanks to CRT [Smart-Vercauteren, ‘14]:

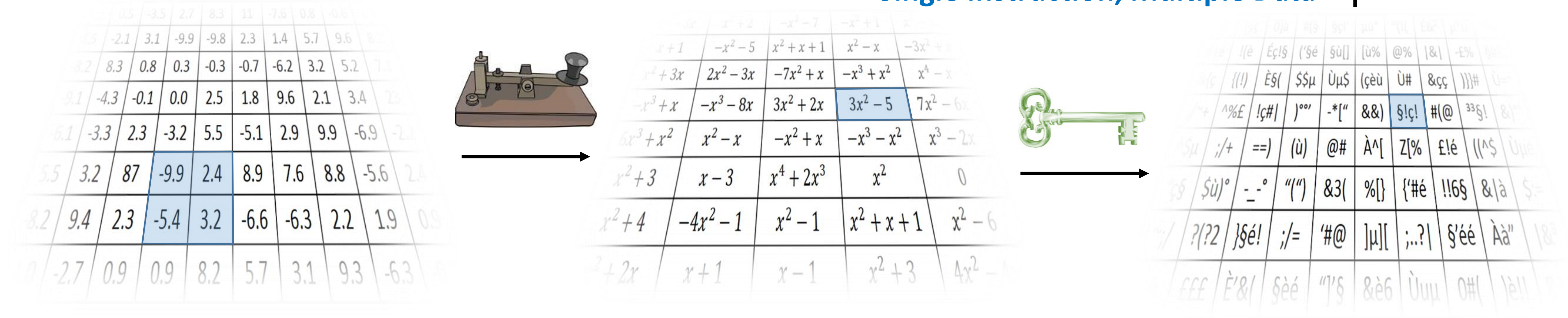
$$R_t = \frac{\mathbf{Z}[x]}{(f(x),t)} \xrightarrow{\cong} \frac{\mathbf{Z}[x]}{(f_1(x),t)} \times \frac{\mathbf{Z}[x]}{(f_2(x),t)} \times \cdots \times \frac{\mathbf{Z}[x]}{(f_r(x),t)}$$

where the $f_i(x)$ are coprime factors of $f(x)$ modulo t.

SIMD

SIMD $\longrightarrow$ $\Big\uparrow\; C_{\text{cryp}}$

Single Instruction, Multiple Data



Batch encoding is possible thanks to CRT [Smart-Vercauteren, '14]:

$$R_t = \frac{\mathbf{Z}[x]}{(f(x),t)} \xrightarrow{\cong} \frac{\mathbf{Z}[x]}{(f_1(x),t)} \times \frac{\mathbf{Z}[x]}{(f_2(x),t)} \times \cdots \times \frac{\mathbf{Z}[x]}{(f_r(x),t)}$$

where the $f_i(x)$ are coprime factors of $f(x)$ modulo t.

Contributions

- SIMD seems incompatible with fractional encoding, because most factors of $x^d + 1$ modulo t are not of that form.

  We give a very general fractional encoding method which does not require that $f(x) = x^d + 1$.

- The CRT allows for more fine-grained decompositions by *also* incorporating factorizations of t.

  We show that this enables more flexible and denser data packing.

Fractional encoding revisited

Write $f(x) = x \cdot g(x) + f(0)$.

First encode

$$a_r b^r + \cdots + a_1 b + a_0 + a_{-1} b^{-1} + \cdots + a_{-s} b^{-s}$$

as a Laurent polynomial in $\mathbf{Z}[x^{\pm 1}]$ by substituting x for b.

Fractional encoding revisited

Write $f(x) = x \cdot g(x) + f(0)$.

mild requirement:
$f(0)$ invertible mod t

First encode

$$a_r x^r + \cdots + a_1 x + a_0 + a_{-1} x^{-1} + \cdots + a_{-s} x^{-s}$$

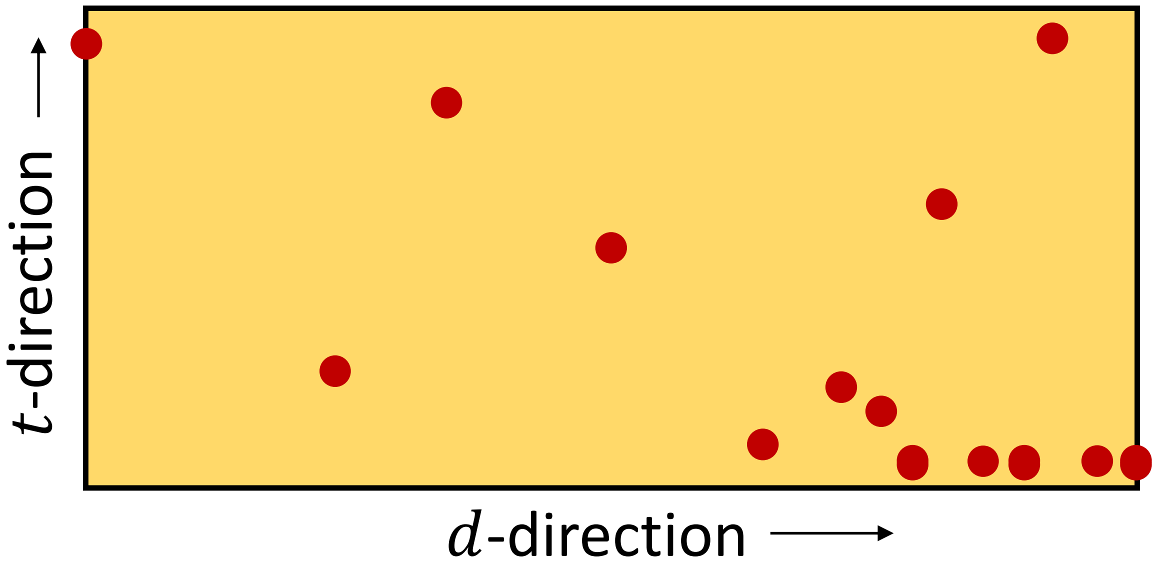
as a Laurent polynomial in $\mathbf{Z}[x^{\pm 1}]$ by substituting x for b.

Then apply:

$$\mathbf{Z}[x^{\pm 1}] \xrightarrow{\bmod t} \mathbf{Z}_t[x^{\pm 1}] \xrightarrow{\eta_f} R_t \text{ where } \eta_f \colon \begin{cases} x \mapsto x \\ x^{-1} \mapsto -f(0)^{-1} g(x) \end{cases}$$

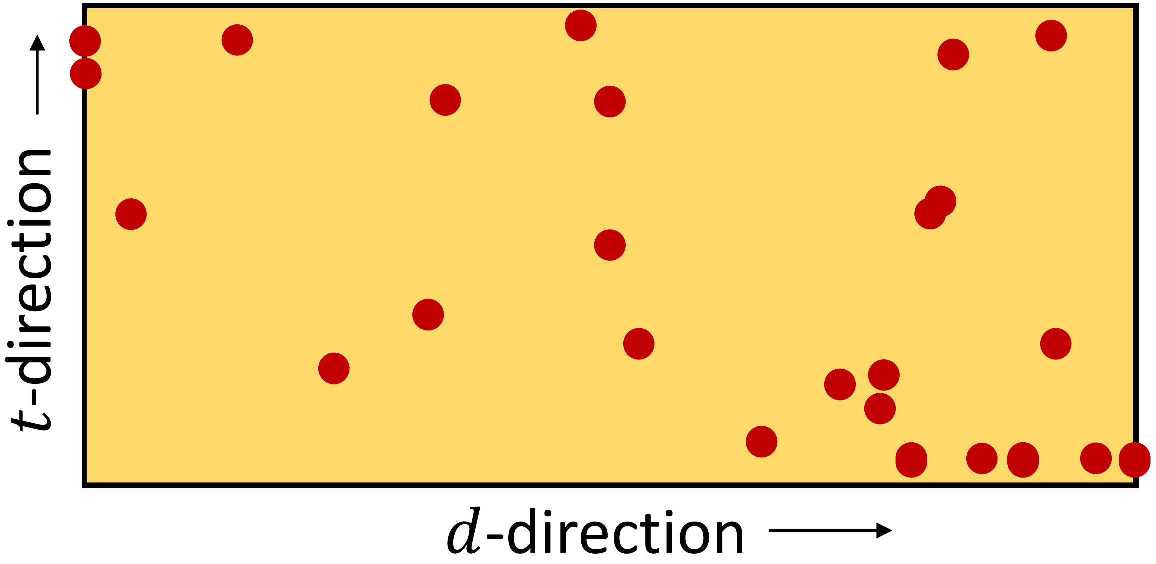
Decoding

Visually: looks like a mess,
seems to overflow from the start!



Decoding

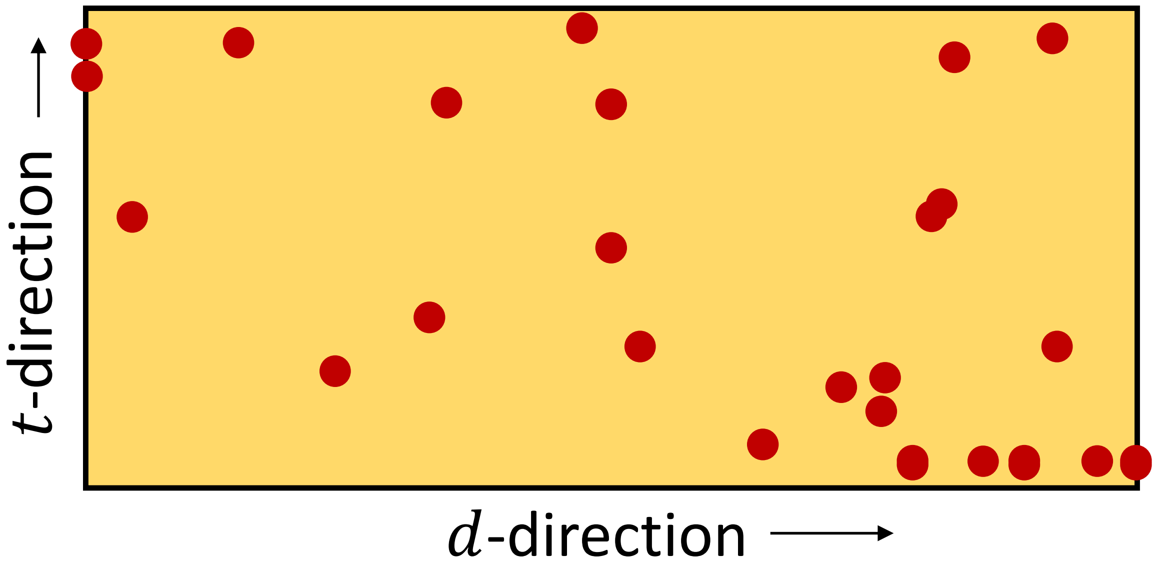
Visually: looks like a mess,
seems to overflow from the start!



Decoding

Visually: looks like a mess,
seems to overflow from the start!

Algebraically, much cleaner.



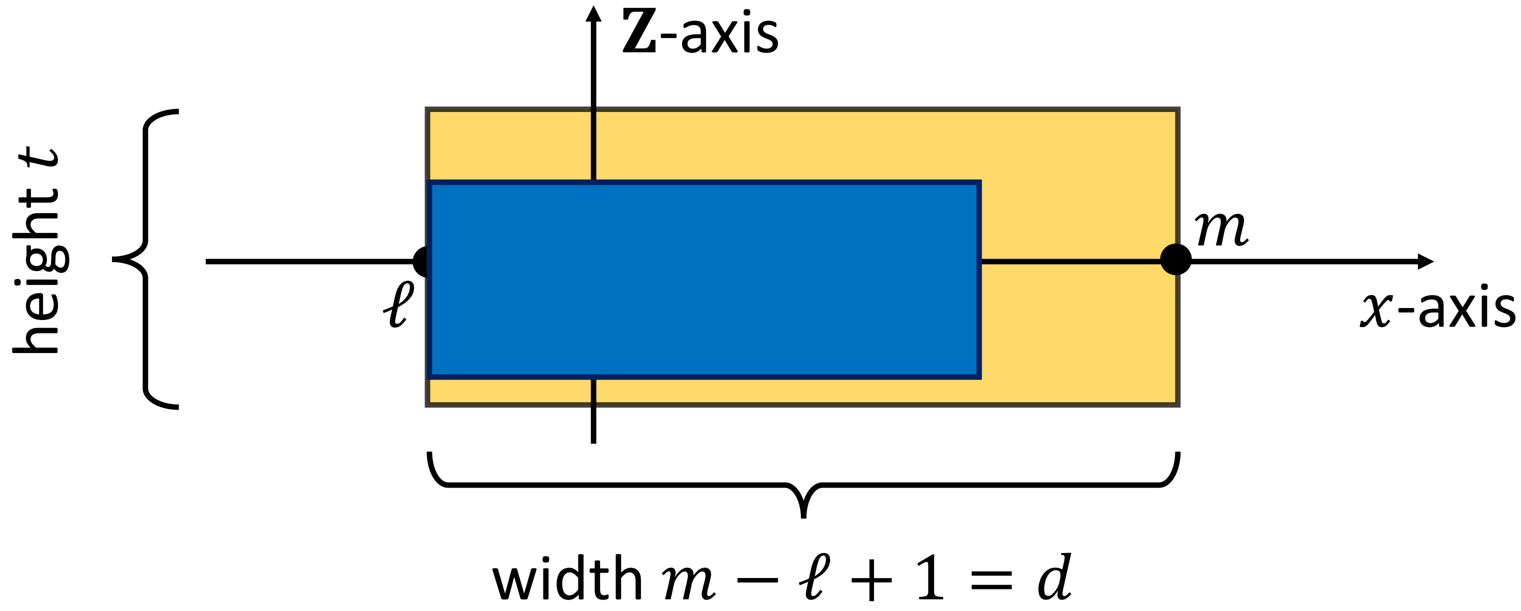
If $m - \ell + 1 = d$ then the restricted map

$$\mathbf{Z}_t[x^{\pm 1}]^{\leq m}_{\geq \ell} \xrightarrow{\eta_f} R_t$$

is an isomorphism of free $\mathbf{Z}_t$-modules of rank d.

Bounding box

Suppose we know that the evaluation of C when carried out in $\mathbf{Z}[x^{\pm 1}]$ ends up in a certain box, and that some shifted *plaintext space covers this box*.



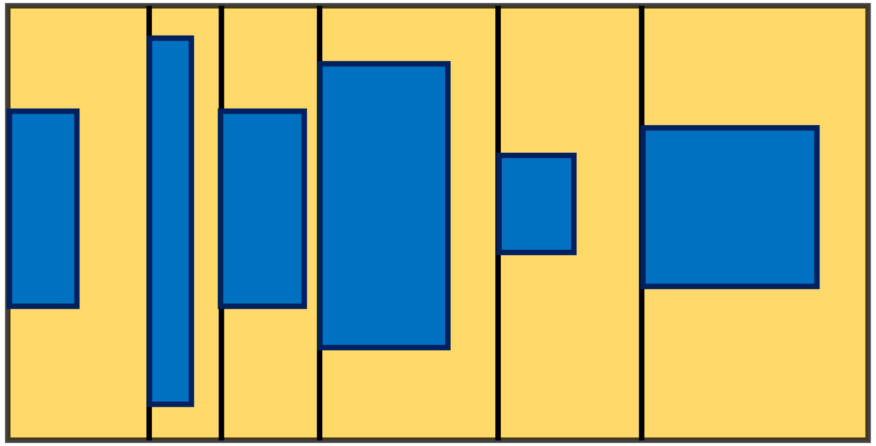
Decoding = inverting $\mathbf{Z}[x^{\pm 1}]^{\leq m}_{\geq \ell} \xrightarrow{\text{mod } t} \mathbf{Z}_t[x^{\pm 1}]^{\leq m}_{\geq \ell} \xrightarrow{\eta_f} R_t$.

Decomposing plaintext space

The CRT decomposition used in [Smart-Vercauteren, ‘14]

$$R_t = \frac{\mathbf{Z}[x]}{(f(x),t)} \xrightarrow{\cong} \frac{\mathbf{Z}[x]}{(f_1(x),t)} \times \frac{\mathbf{Z}[x]}{(f_2(x),t)} \times \cdots \times \frac{\mathbf{Z}[x]}{(f_r(x),t)}$$

can be viewed as a vertical slicing of plaintext space:



Each individual slice should cover the bounding box of the corresponding entry.

Decomposing plaintext space

We generalize this discussion: suppose

$$t = t_1 t_2 t_3 \cdots t_s \qquad \text{and} \qquad f(x) = \prod_{i=1}^{r_i} f_{ij}(x) \mod t_i$$

are factorizations into coprimes. Then:

$$R_t = \frac{\mathbf{Z}[x]}{(f(x), t)} \cong \frac{\mathbf{Z}[x]}{(f(x), t_1)} \times \\ \vdots \\ \times \frac{\mathbf{Z}[x]}{(f(x), t_s)}$$

Decomposing plaintext space

We generalize this discussion: suppose

$$t = t_1 t_2 t_3 \cdots t_s \qquad \text{and} \qquad f(x) = \prod_{i=1}^{r_i} f_{ij}(x) \mod t_i$$

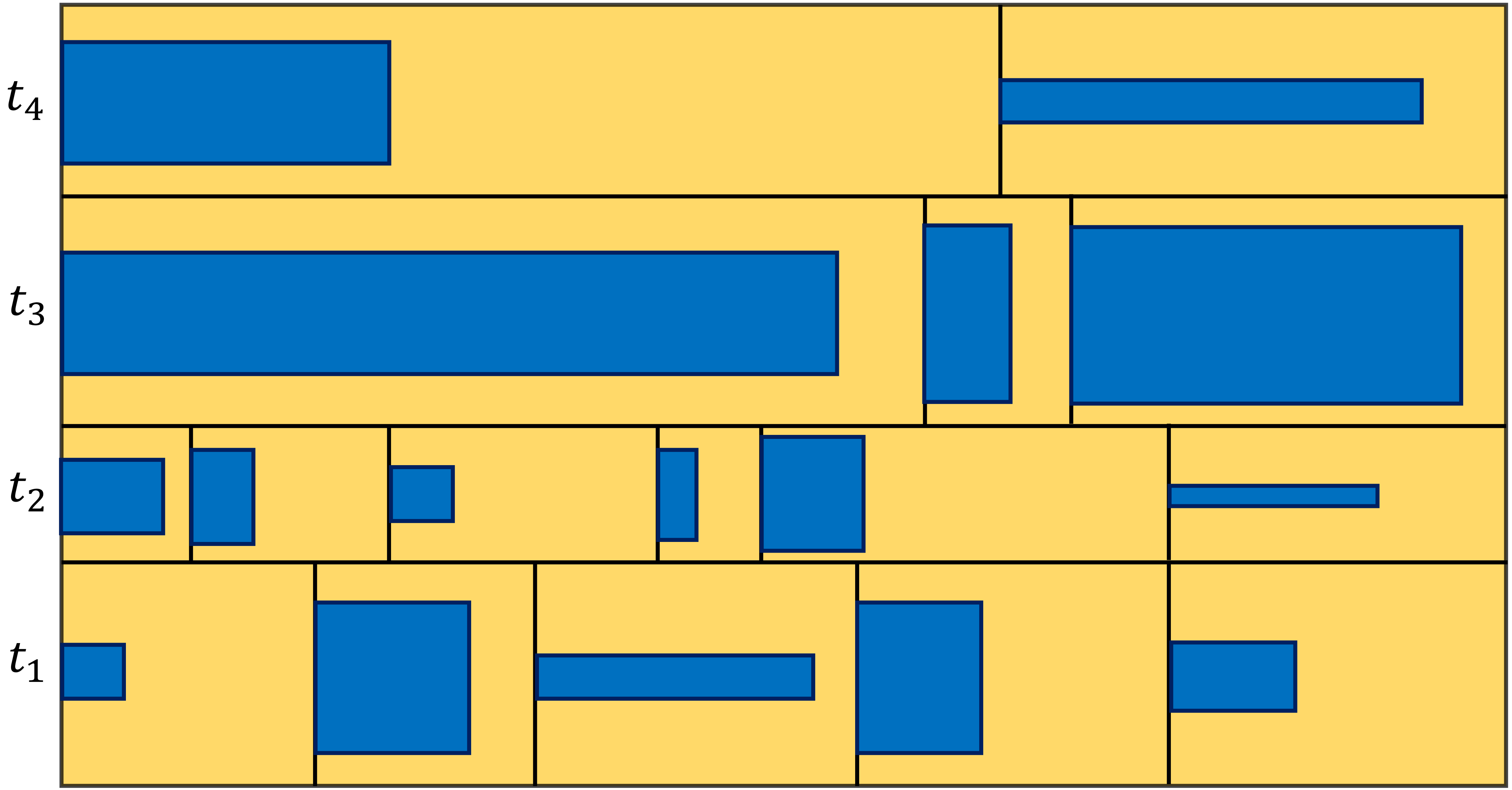
are factorizations into coprimes. Then:

$$R_t = \frac{\mathbf{Z}[x]}{(f(x), t)} \cong \frac{\mathbf{Z}[x]}{(f_{11}(x), t_1)} \times \frac{\mathbf{Z}[x]}{(f_{12}(x), t_1)} \times \cdots \times \frac{\mathbf{Z}[x]}{(f_{1r_1}(x), t_1)} \times$$

$$\vdots$$

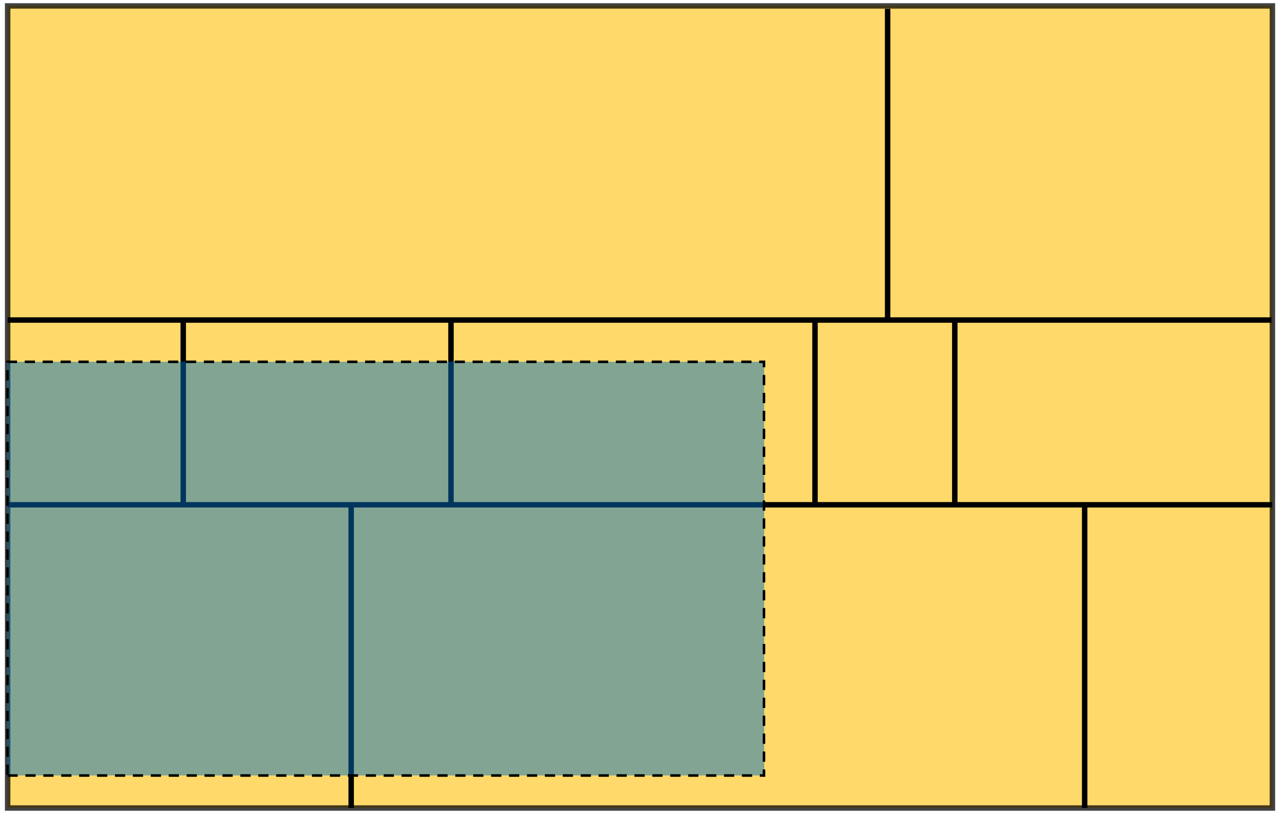
$$\times \frac{\mathbf{Z}[x]}{(f_{s1}(x), t_s)} \times \frac{\mathbf{Z}[x]}{(f_{s2}(x), t_s)} \times \cdots \times \frac{\mathbf{Z}[x]}{(f_{sr_s}(x), t_s)}$$

Decomposing plaintext space



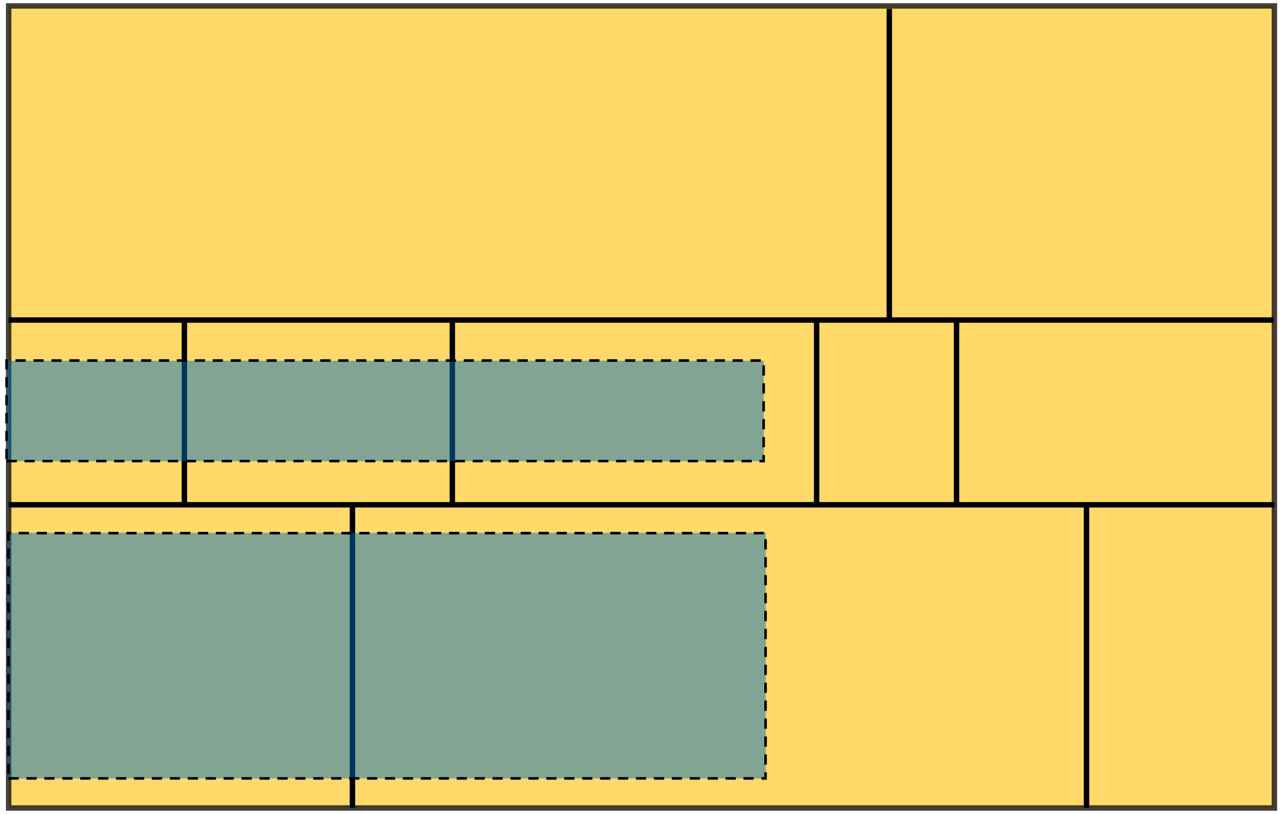
Blocks

What if a computation does not fit into one of these *bricks*?



Blocks

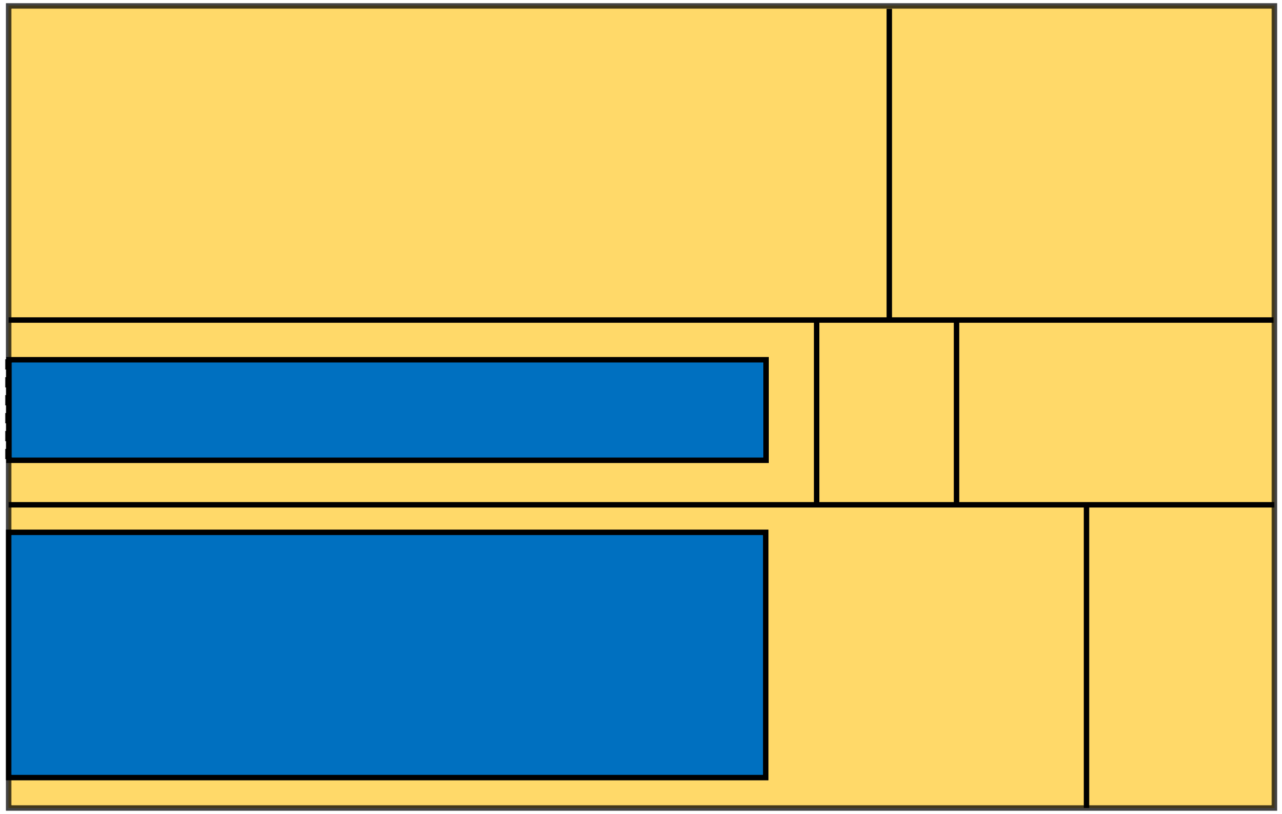
What if a computation does not fit into one of these *bricks*?



Distribute computation over enough horizontal slices.

Blocks

What if a computation does not fit into one of these *bricks*?



Distribute computation over enough horizontal slices.

In each horizontal slice, select enough factors $f_{ij}(x)$.

Gives rise to notion of *block*:

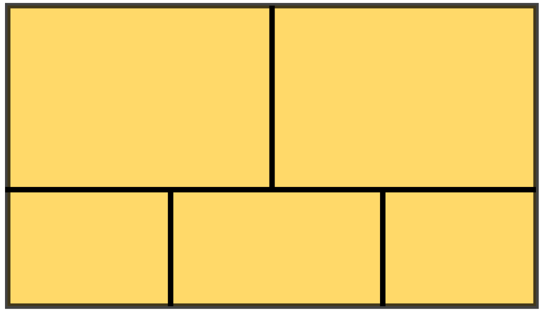
$$\bigcup_{i\in I}\bigcup_{j\in J_i}\left\{\left(t_i, f_{ij}(x)\right)\right\}$$

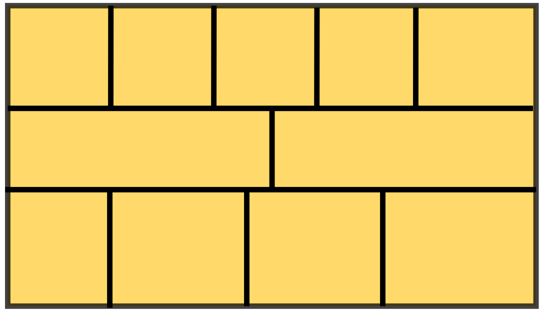
Toolkit for optimal packing

Choose good t for given circuit C and dataset, taking into account:

- lower bounds coming from correct decoding,
- upper bound coming from correct decryption,
- splitting behaviour:

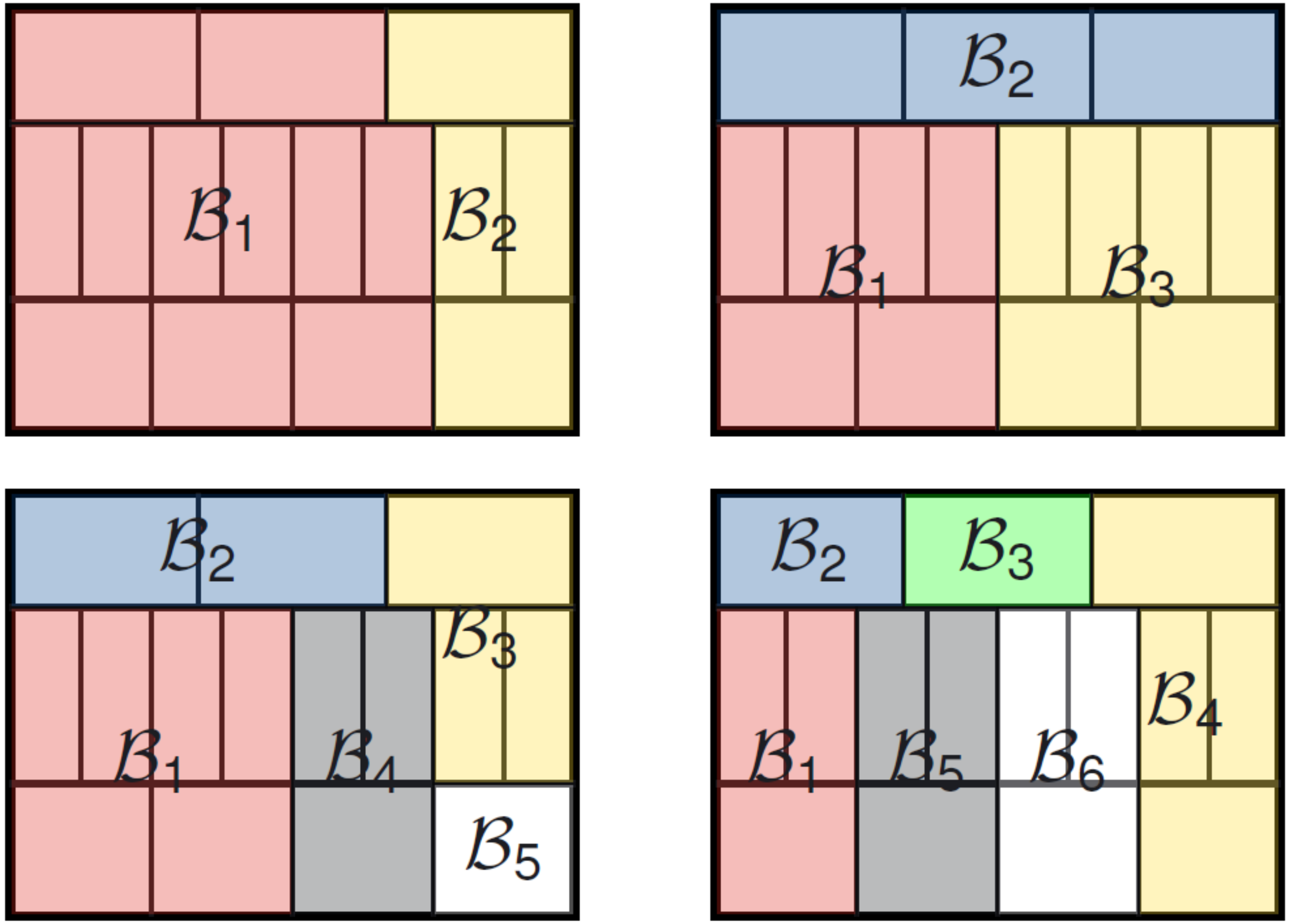
  similar-sized t's give very different brick structures.





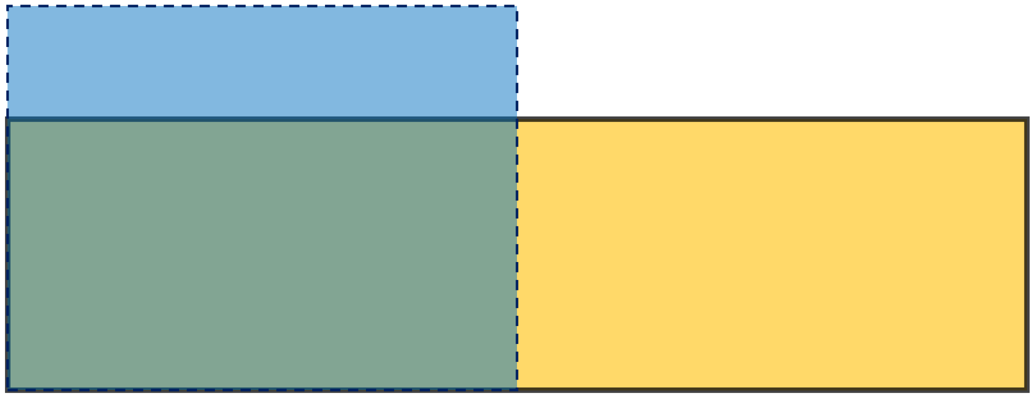
Toolkit for optimal packing

Choose set of blocks that make the best fit for the computation.



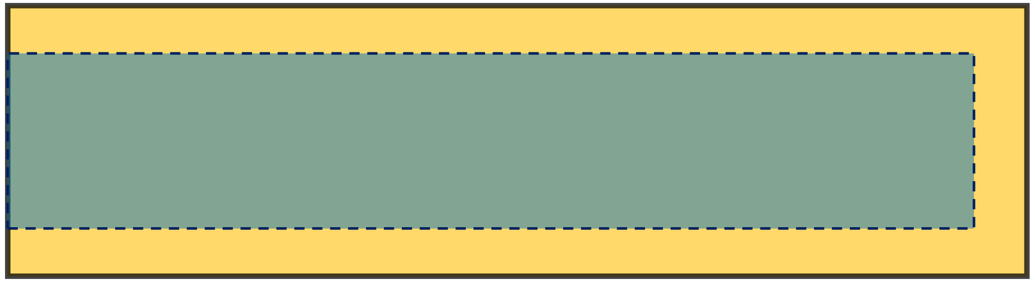
Toolkit for optimal packing

Choose appropriate encoding base b, can be specific to block.



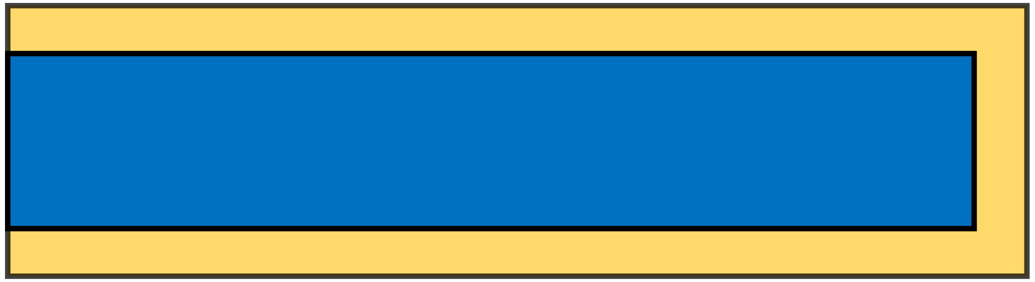
Toolkit for optimal packing

Choose appropriate encoding base b, can be specific to block.

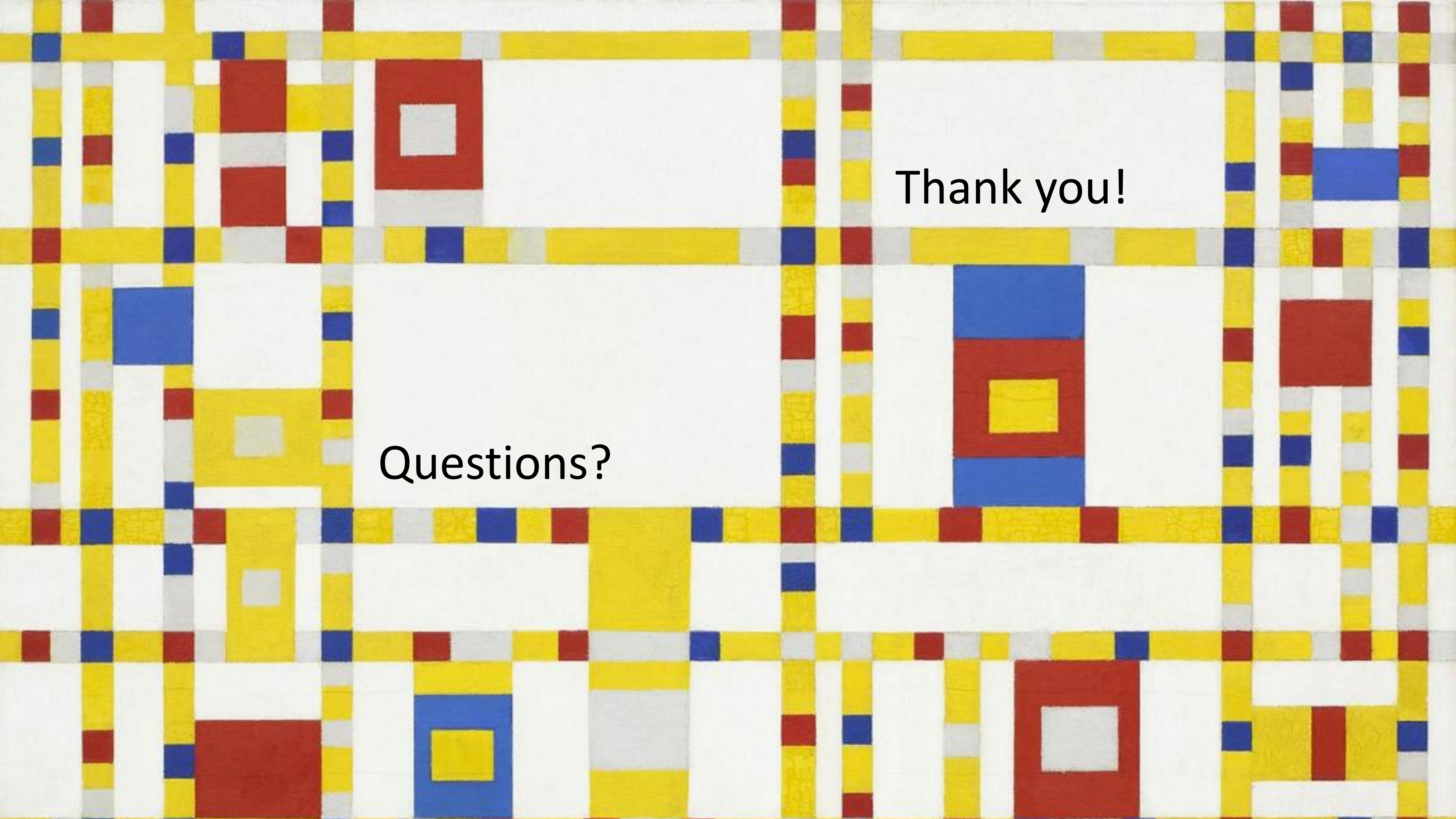


Toolkit for optimal packing

Choose appropriate encoding base b, can be specific to block.



Smaller base gives wider but lower encodings.



Thank you!

Questions?