mCarve: Carving attributed dump sets

Sjouke Mauw
University of Luxembourg
sjouke.mauw@uni.lu
http://satoss.uni.lu/sjouke/

(joint work with Ton van Deursen, Saša Radomirović)

Public transportation cards easily hacked

Luxembourg: e-go card

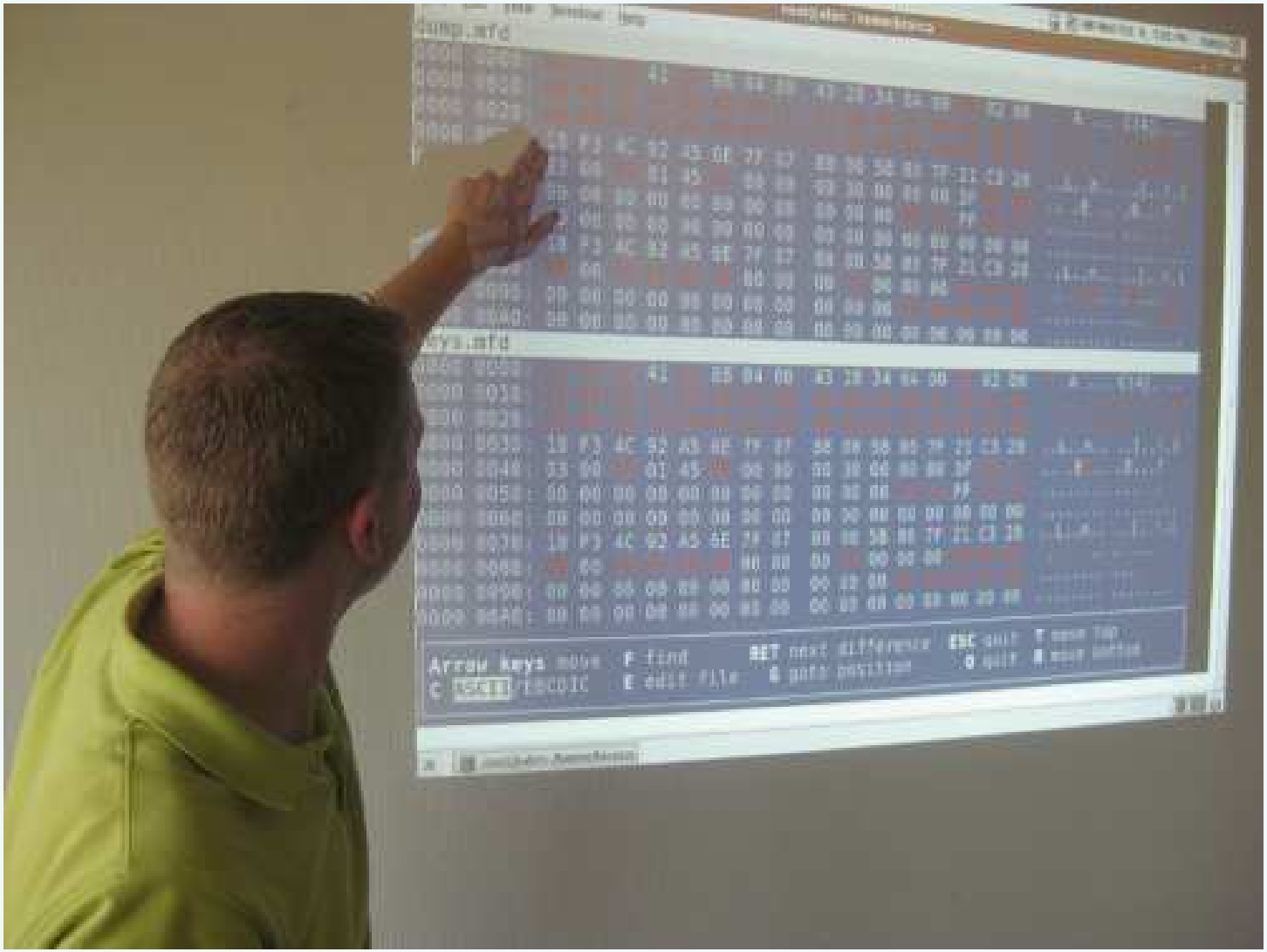
All you need is. . .

. . . a reader, a laptop, publicly available software, a Ton.

But decrypting the card is just the first step

“What do all these bits and bytes mean?”

Manual analysis needed



“Is the *number-of-rides-left* stored here?”

Manual analysis is labour intensive

“Hmm, not sure about that.”

Existing problem from digital forensics

Carving = recover data from a memory dump of a device

Our problem is different

1. Not one single dump, but a series of dumps.
2. For every dump we know some attributes, e.g.
   - card “identity”,
   - date-of-purchase,
   - type-of-card,
   - rides-left,
   - time-of-use.

Standard carving tools don’t apply

Research question

Develop a methodology to answer:

- Are these attributes encoded in the dumps?
- Where?
- With which encoding?

Assumptions:

1. All dumps of same length.
2. Attributes are stored at the same location in every dump.
   (can be relaxed)
3. Encoding of attribute is deterministic and injective.

Central notion: attribute mapping

- $a \in \mathbb{A}$ an attribute (e.g. *rides-left*)
- $s \in \mathbb{B}^n$ a dump (i.e. a bit string of length n)
- $S \subseteq \mathbb{B}^n$ a dump set
- $s|_I$ substring of dump s, restricted to $I \subseteq [0, n)$
- $val_a(s)$ the value of attribute a for dump s
  (e.g. $val_{\textit{rides-left}}(s) = 5$)
- $e(val_a(s))$ an injective encoding of the value of attribute a as a bit string
  (e.g. 5 is encoded as 0101)

Central notion: attribute mapping

- $a \in \mathbb{A}$ an attribute (e.g. *rides-left*)
- $s \in \mathbb{B}^n$ a dump (i.e. a bit string of length n)
- $S \subseteq \mathbb{B}^n$ a dump set
- $s|_I$ substring of dump s, restricted to $I \subseteq [0, n)$
- $val_a(s)$ the value of attribute a for dump s
  (e.g. $val_{\textit{rides-left}}(s) = 5$)
- $e(val_a(s))$ an injective encoding of the value of attribute a as a bit string
  (e.g. 5 is encoded as 0101)

An attribute mapping determines for every attribute the bit positions where the attribute is stored.

An *attribute mapping* for S is a function $f : \mathbb{A} \to \mathcal{P}([0, n))$, such that for all $a \in \mathbb{A}$ there exists an encoding e with

$$\forall_{s \in S} \quad s|_{f(a)} = e(val_a(s)).$$

Research question formalized

Given a set of dumps $s \in S$ and a set of attributes $a \in \mathbb{A}$ and their values $val_a(s)$, find all possible attribute mappings f.

Example

Finding the *rides-left* attribute.

	rides-left	dump
s_1	4	01010010011101000010 0
s_2	4	00110010000101001011 0
s_3	5	10111010101101010001 1
s_4	6	00101011011101101101 1
s_5	6	11101011001101100110 0

Example

Finding the *rides-left* attribute.

	rides-left	dump	encoding
s_1	4	01010***0100***111***0100***00100	0100
s_2	4	00110***0100***001***0100***10110	0100
s_3	5	10111***0101***011***0101***00011	0101
s_4	6	00101***0110***111***0110***11011	0110
s_5	6	11101***0110***011***0110***01100	0110

Two possibilities for this encoding:

- $f(\textit{rides-left}) = [5, 8]$
- $f(\textit{rides-left}) = [12, 15]$

Example

Finding the *rides-left* attribute.

	rides-left	dump	encoding
s_1	4	010*1001*00111010000100	1001
s_2	4	001*1001*00001010010110	1001
s_3	5	101*1101*01011010100011	1101
s_4	6	001*0101*10111011011011	0101
s_5	6	111*0101*10011011001100	0101

And for another encoding

- $f(\textit{rides-left}) = [3, 6]$

Observations

- **Commonalities**:
  If two dumps have the same attribute value, then the dumps must be identical at the positions of $f(a)$.
- **Dissimilarities**:
  If two dumps have a different attribute value, then the dumps differ in at least one bit at the positions of $f(a)$.

Idea:

Use this to restrict the search for attribute mappings, independently of the encoding.

1. Commonalities

A *bundle* is a collection of dumps with the same attribute value.

$$bundles(a, S) = \{\{s \in S \mid val_a(s) = d\} \mid d \in type(a)\}$$

The *common set* determines which bits in the dumps of a dump set are equal if the attribute values are equal.

$$common(a, S) = \bigcap_{b \in bundles(a,S)} \{i \in [0, n) \mid \forall_{s,s' \in b}\ s_i = s'_i\}.$$

Example: common set

Determine common set (*) per bundle and combine.

	rides-left	dump
s_1	4	01010010011101000010 0
s_2	4	00110010000101001011 0
		*..******..*****.**.*
s_3	5	10111010101101010001 1
		*********************
s_4	6	00101011011101101101 1
s_5	6	11101011001101100110 0
		..*******.******.*...

Example: common set

Determine common set (*) per bundle and combine.

	rides-left	dump
s_1	4	01010010011101000010 0
s_2	4	00110010000101001011 0
		*..******..*****.**.*
s_3	5	10111010101101010001 1
		*********************
s_4	6	00101011011101101101 1
s_5	6	11101011001101100110 0
		..*******.******.*...
common		...******..*****.*...

Conclusion: rides-left must be encoded within the *-ed bits.

Example: common set

Determine common set (*) per bundle and combine.

	rides-left	dump
s_1	4	01010010011101000010 0
s_2	4	00110010000101001011 0
		*..******..*****.**.*
s_3	5	10111010101101010001 1
		*********************
s_4	6	00101011011101101101 1
s_5	6	11101011001101100110 0
		..*******.******.*...
common		...******..*****.*...

Conclusion: rides-left must be encoded within the *-ed bits.

Complexity: $O(n \cdot |S|)$

2. Dissimilarities

The *dissimilarity set* contains all subsets I of $[0, n)$ such that if the attribute value of any pair of dumps differs, I has a bit that differs.

$$dissim(a, S) = \{I \subseteq [0, n) \mid \forall_{s,s' \in S}(val_a(s) \neq val_a(s') \implies \exists_{i \in I} s_i \neq s'_i)\}$$

We can optimize this by taking one representative of each bundle.

Example: dissimilarity set

	rides-left	dump
s_1	4	*01*01001001110100001 00
s_3	5	*10*11101010110101000 11
s_4	6	*00*10101101110110110 11
		** . . . . .

Example: dissimilarity set

	rides-left	dump
s_1	4	0*101*00100111010000100
s_3	5	1*011*10101011010100011
s_4	6	0*010*10110111011011011
		..... .*.....

Example: dissimilarity set

	rides-left	dump
s_1	4	01010010011101000010 0
s_3	5	10111010101101010001 1
s_4	6	00101011011101101101 1
		..... .*..... ..**..... ...**..... ....****..... .....****..... ......***..... ....**..... etc.

Example: dissimilarity set

	rides-left	dump
s_1	4	01010010011101000010 0
s_3	5	10111010101101010001 1
s_4	6	00101011011101101101 1
		..... .*..... ..**..... ...**..... ....****..... .....****..... ......***..... ....**..... etc.

Conclusion: the encoding of rides-left must include at least one of the starred intervals.

Example: dissimilarity set

	rides-left	dump
s_1	4	01010010011101000010 0
s_3	5	10111010101101010001 1
s_4	6	00101011011101101101 1
		..... .*..... ..**..... ...**..... ....****..... .....****..... ......***..... ....**..... etc.

Conclusion: the encoding of rides-left must include at least one of the starred intervals.

Complexity: $O(n^2\,|S| + n\,|S|\log|S|)$

Main theorem

Let $\mathbb{A}$ be an attribute set and let f be an attribute mapping for dump set $S \subseteq \mathbb{B}^n$, then

$$\forall_{a \in \mathbb{A}} \exists_{I \in dissim(a,S)} \; I \subseteq f(a) \subseteq common(a, S).$$

Example: common + dissim

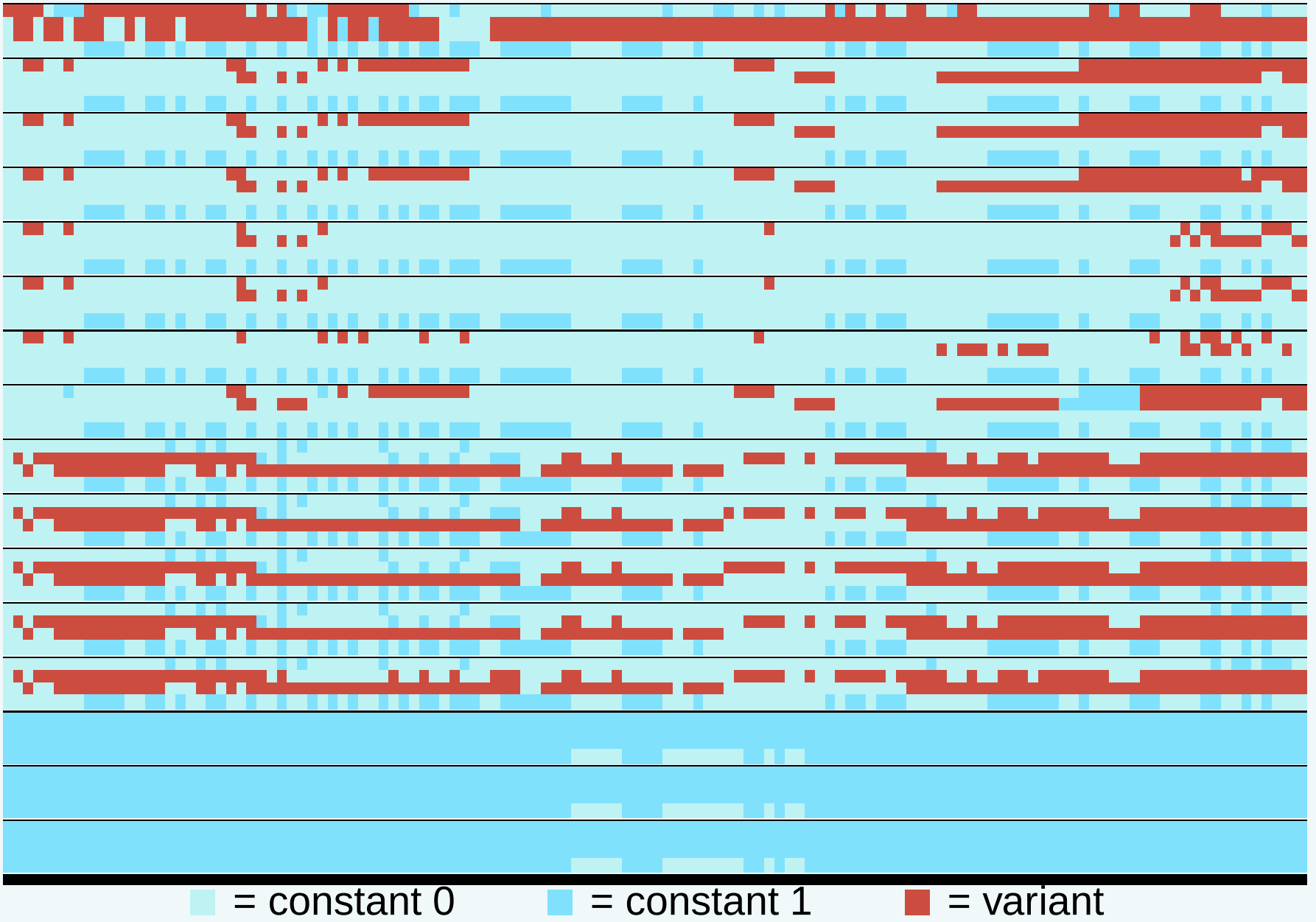
Assuming 4 bits, 4 remaining possibilities.

	rides-left	dump
s_1	4	01010010011101000010 0
s_2	4	00110010000101001011 0
s_3	5	10111010101101010001 1
s_4	6	00101011011101101101 1
s_5	6	11101011001101100110 0
		...****..... ....****..... .....****..... .....****.....

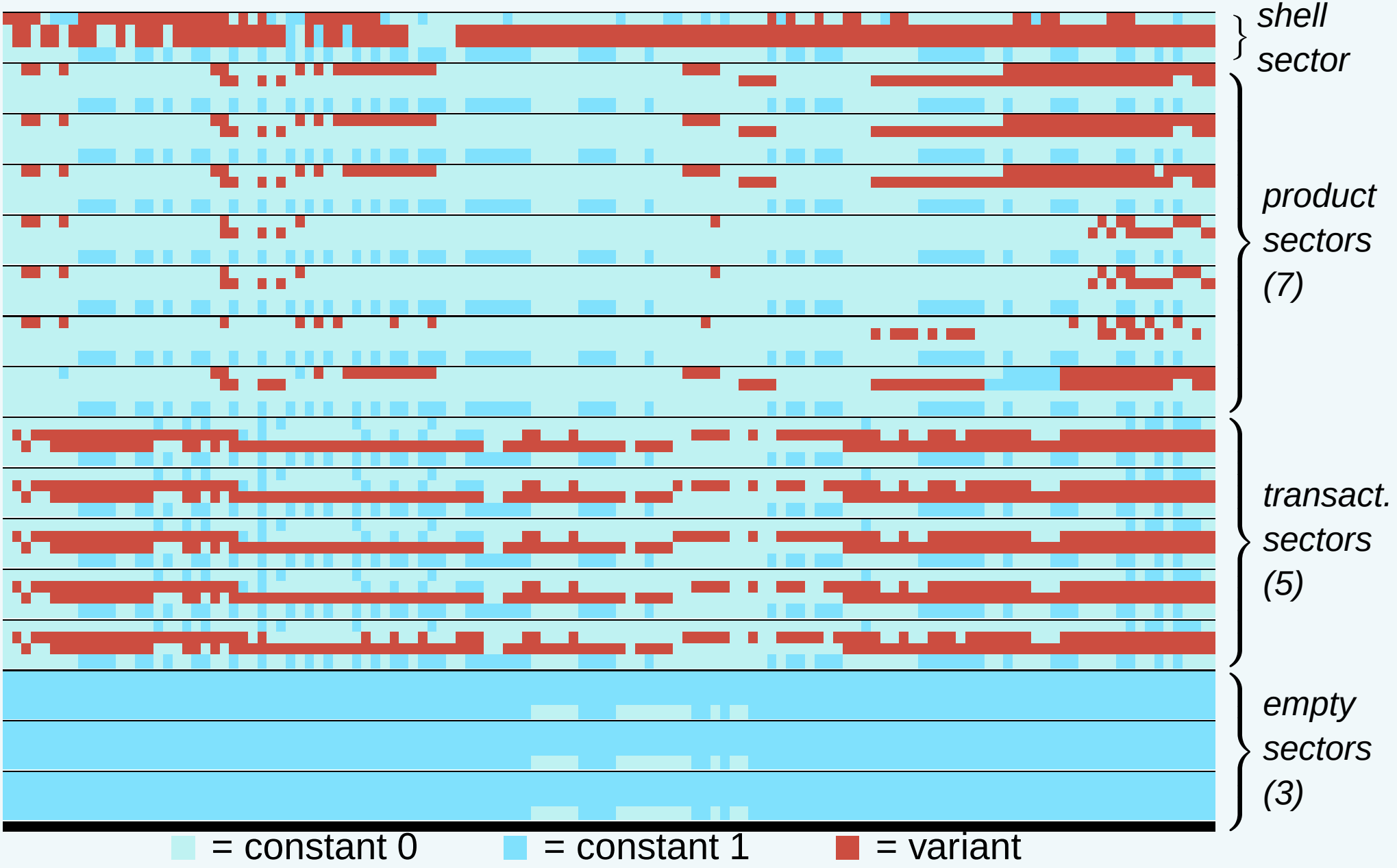
Application: e-go card

- Developed prototype tool.
- Collected 68 dumps from 7 cards.
- Wrote down attributes for each dump: rides-left, card-type, license-plate, swipe-time, swipe-date, etc.

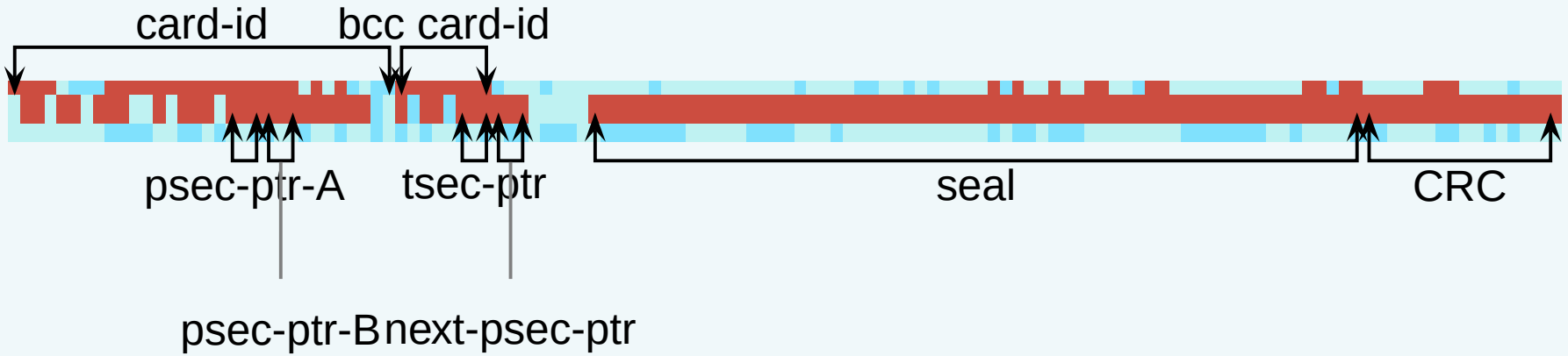
Applying “common”



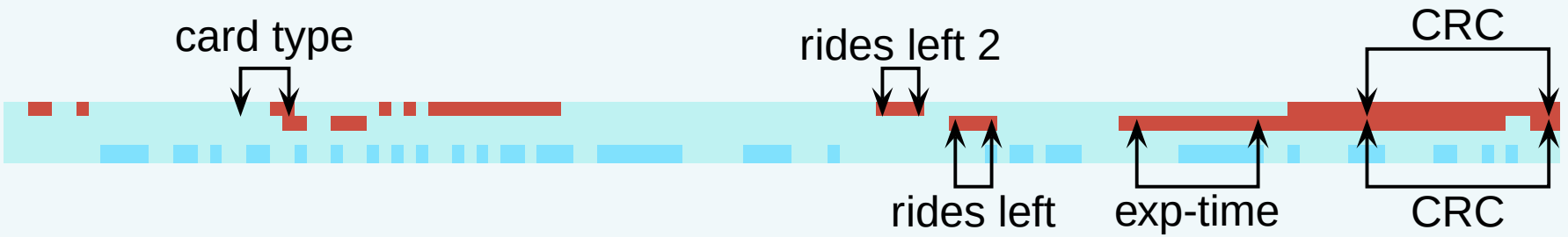
Applying “common”



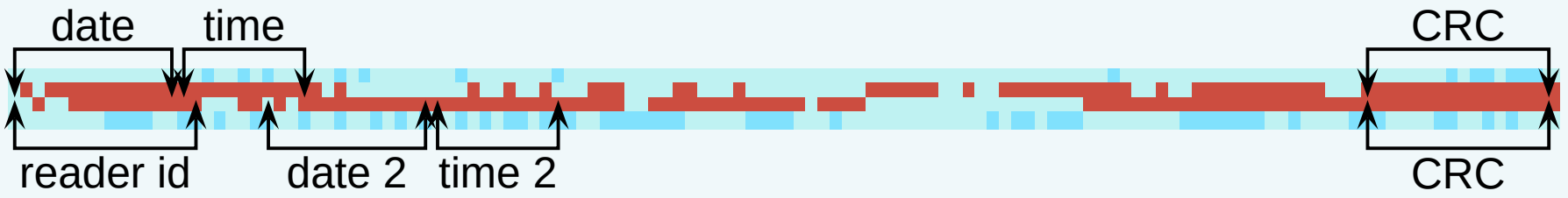
Shell sector



Product sector



Transaction sector



Conclusion

- We defined the *carving problem* for attributed dump sets.
- Developed algorithms and prototype tool.
- Results for e-go card: can find most attributes we collected.
- Can also find “internal” and “semi-static” attributes.
- Performance: few seconds for e-go dump set.
- Convergence: need approximately 10 bundles to find a regular attribute.
- Future work:
  - automatically recover encoding
  - develop “attribute algebra”
  - algorithms to improve robustness
  - application to security protocol reengineering
  - recode prototype in C

Download prototype tool from:

http://satoss.uni.lu/mcarve/