Practical Solutions for Format Preserving Encryption

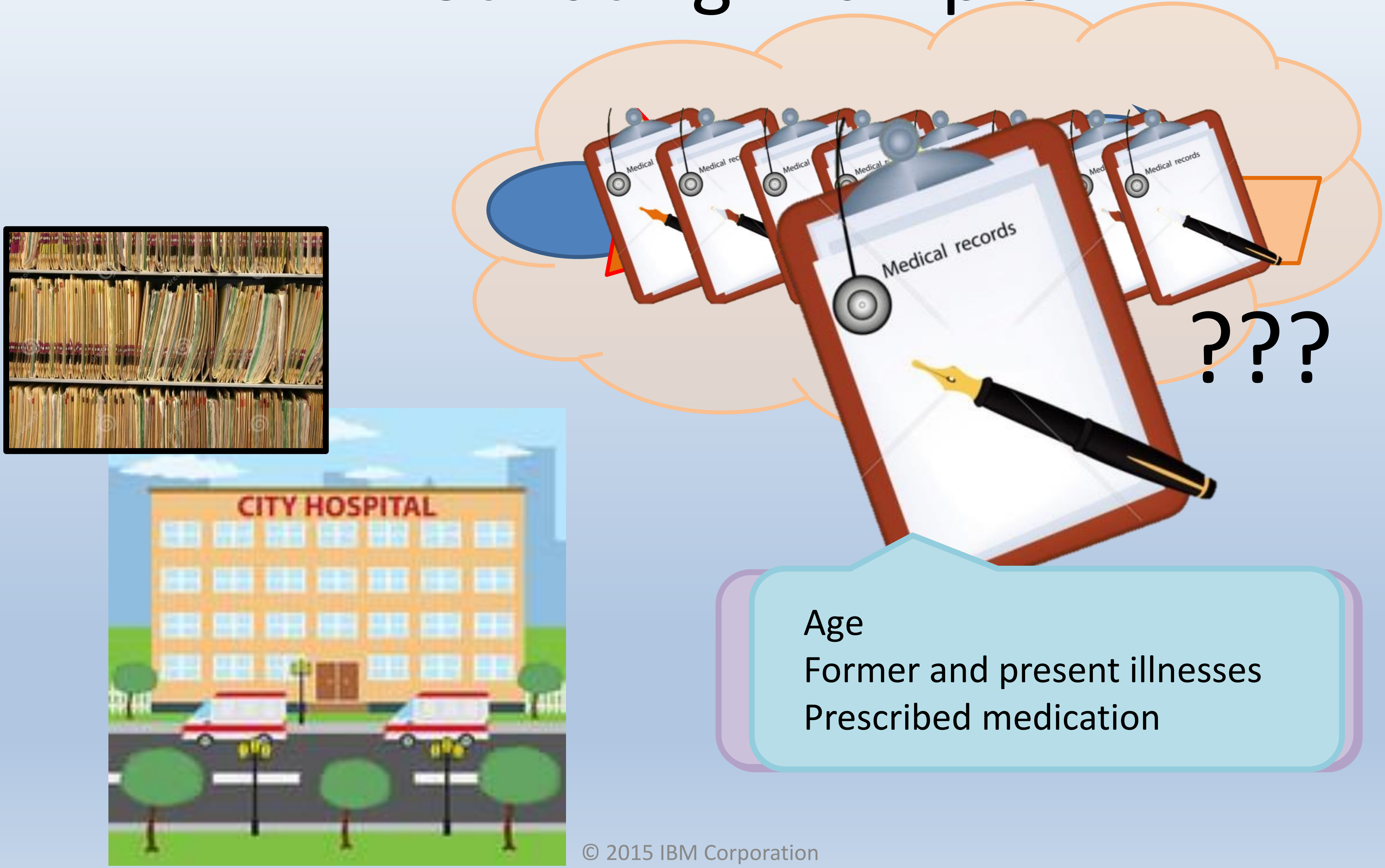
Authors: Mor Weiss, Boris Rozenberg, and Muhammad Barham
Presented by Boris Rozenberg (borisr@il.ibm.com)

May 21, 2013

Talk Outline

- Motivating example
- Encryption: background
- Format Preserving Encryption (FPE):
  - Simple constructions
  - Better constructions:
    - Representing general formats
    - Encrypting general formats
  - Dealing with large formats
  - Evaluation
- Concurrent Work
- Conclusion

Motivating Example



Age
Former and present illnesses
Prescribed medication

Encryption

(keeping data private)

Encryption Schemes

- A triplet $\Pi = (KeyGen, Enc, Dec)$ of algorithms
  - Π associated with 3 sets:
    - $\mathcal{K}$: domain of valid keys
    - $\mathcal{M}$: message domain
    - $\mathcal{C}$: ciphertext domain.
  - $KeyGen$ generates random key from $\mathcal{K}$
  - Enc on message (plaintext) $m \in \mathcal{M}$ and key $k \in \mathcal{K}$ outputs ciphertext $c \in \mathcal{C}$
  - Dec on ciphertext $c \in \mathcal{C}$ and key $k \in \mathcal{K}$ outputs message $m \in \mathcal{M}$
- Deterministic encryption: only $KeyGen$ is randomized
  - Everything deterministic once key is chosen
- Assumed adversary knows everything but key

Encryption Schemes: Required Properties

- A triplet $\Pi = (KeyGen, Enc, Dec)$ of algorithms
- Correctness: for every $k \in \mathcal{K}$ and every $m \in \mathcal{M}$

$$Dec\big(k, Enc(k, m)\big) = m$$

- Security:
  - Many security notions
  - Intuitively, ciphertext c reveals (almost) no information on message m
    - Even if adversary has prior knowledge
  - Achieved by random 1:1 functions
- For usability, all algorithms must be efficient

Security-Efficiency Tradeoffs

Efficiency

$Enc(k, m) = m$ for every key k

Security

$Enc(k,\cdot)$ applies a random 1:1 function

Format Preserving Encryption

(encrypting to “acceptable” formats)

Format Preserving Encryption (FPE)

- Standard encryption maps messages to “garbage”
  - May be impossible to store ciphertext in same tables
  - Applications using data may crash
- Need some plaintext properties to be preserved
- FPE: *Deterministic* encryption scheme $\Pi = (KeyGen, Enc, Dec)$
- with additional property $\mathcal{M} = \mathcal{C}$
- Ciphertexts have the same format as plaintexts!
  - Social security number (ssn) mapped to legal ssn
  - Credit card number (ccn) mapped to legal ccn
  - Address mapped to legal address
  - Etc…

Example: The DES Encryption

[redacted]

64-bit string

DES is format-preserving!

64-bit string

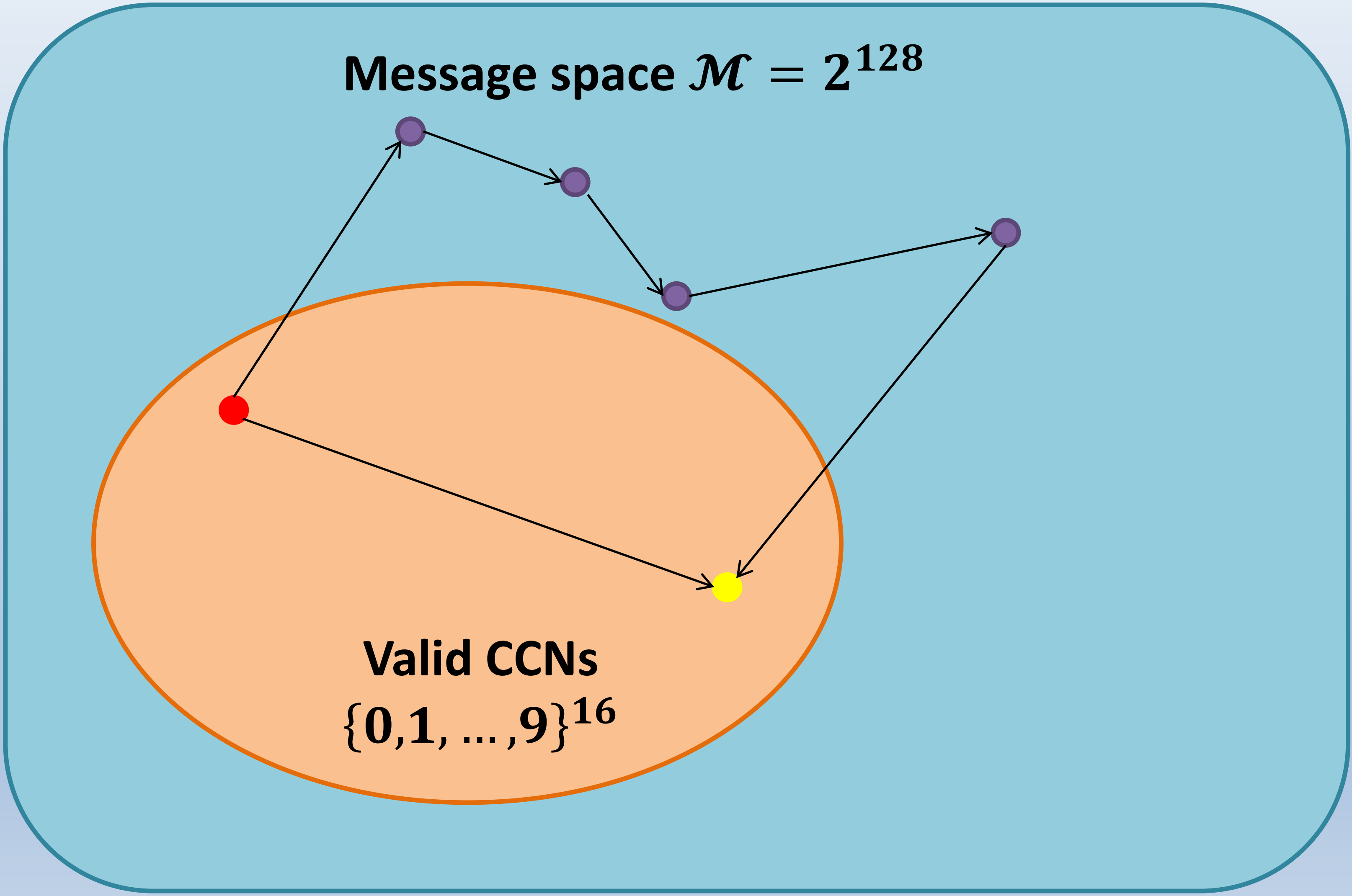
FPE Schemes For General Formats: Simple Solution

- Known encryption schemes are FP for *fixed, specific* formats
  - Usually, bit strings of fixed length
- What about other formats?
  - For CCNs, message space $\subseteq \{0,1,\ldots,9\}^{16}$
  - No known encryption for this message space!
- Can use cycle-walking [Black-Rogaway'02]

"if at first you don't succeed, you pick yourself up and try again"

  - Use "standard" encryption with $\{0,1,\ldots,9\}^{16} \subseteq \mathcal{M}$
  - Repeat until ciphertext in $\{0,1,\ldots,9\}^{16}$

Cycle-Walking



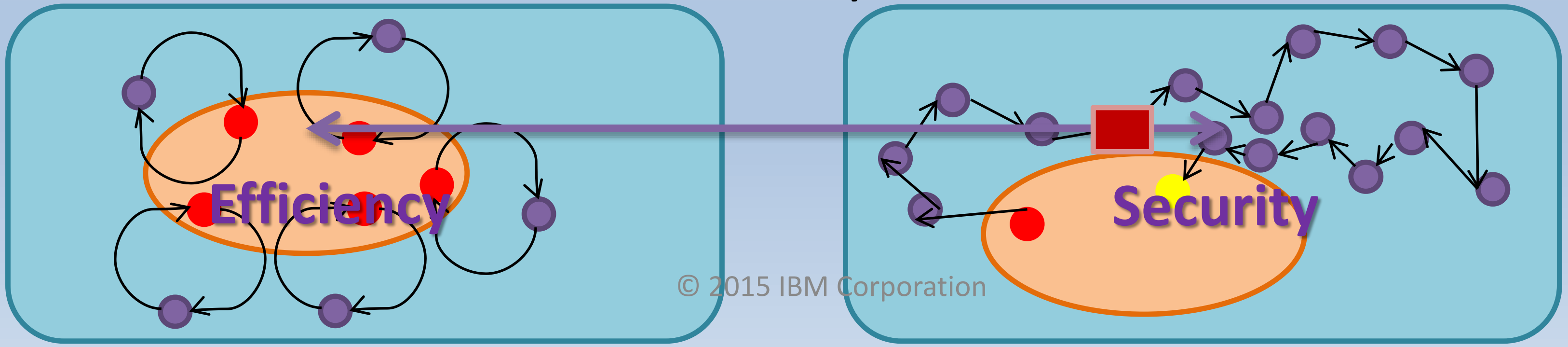
Message space $\mathcal{M} = 2^{128}$

Valid CCNs

$\{0,1,\ldots,9\}^{16}$

Cycle-Walking: Pros and Cons

- Pros:
  - Use “off-the-shelf” encryption schemes
    - One design for all formats
  - Known encryption schemes are provably secure
- Cons:
  - *Average* efficiency depends on ratio between format-size and message domain size
    - Need to repeat $\frac{format\ size}{|\mathcal{M}|}$ times on average
  - No bound on actual efficiency



Improved FPEs for Numeric Domains

- Several known schemes for numeric domains
  - Considered due to (in)efficiency of cycle walking
- [Bellare et al. '09] construct integer-FPE: FPE with $\mathcal{M} = \{0,1,\dots,M-1\}$

What about non-numeric domains?

From Integer-FPE to General-Format FPE

- Can base general-format FPE on integer-FPE using Rank-then-Encipher (RtE): [Bellare et al. ’09]
  - Message space $\mathcal{M}$ arbitrarily ordered: $\text{rank}: \mathcal{M} \rightarrow \{0,1,..,M$

Warm-Up Example

X	*y*	7	*A*
upper case	lower case	digit	upper case

idea: compute location in lexicographic order

index each character

23	**24**	**7**	**0**

rank calculated by scaling and summing the indices

$23 \cdot 26 \cdot 10 \cdot 26 \quad + \quad 24 \cdot 1$ $\cdot 26 \quad + \quad 0$

Sum-and-Scale !

ge g method

$$1234 = 1 \cdot 10 \cdot 10 \cdot 10 \quad + \quad 2 \cdot 10 \cdot 10 \quad + \quad 3 \cdot 10 \quad + \quad 4$$

Ranking General Formats: Simple Solution

- Want: ***efficient*** rank: $\mathcal{M} \to \{0,1,..,M-1\}$
- Can rank every format $\mathcal{F}$ defined by
  - Length ℓ
  - Sets $\Sigma_1, \dots, \Sigma_\ell$ of “legal” characters in locations $1, \dots, \ell$.
- Simple solution:
  - Divide $\mathcal{M}$ to subsets $\mathcal{M}_1, \dots, \mathcal{M}_k$
  - $\mathcal{M}_i$ defined by $\ell_i, \Sigma_1^i, \dots, \Sigma_{\ell_i}^i$ **How to define efficiently?!**
  - Rank and encryption of $m \in \mathcal{M}_\mathrm{i}$ computed in relation to $\mathcal{M}_i$

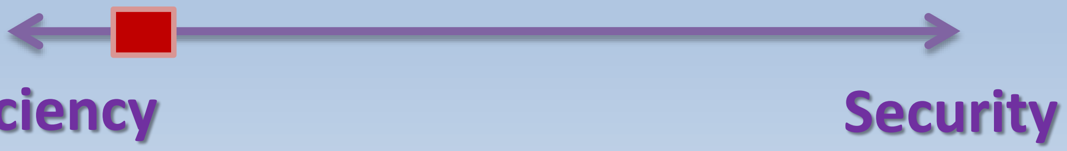
Simple Solution: Security Analysis

Simple solution:

- Divide $\mathcal{M}$ to subsets $\mathcal{M}_1, \dots, \mathcal{M}_k$
- $\mathcal{M}_i$ defined by $\ell_i \Sigma_1^i, \dots, \Sigma_\ell^i$
- Rank and encryption of $m \in \mathcal{M}_\mathrm{i}$ computed in relation to $\mathcal{M}_i$

Security is compromised:

- Ranking computed in every $\mathcal{M}_i$ separately
- So $m \in \mathcal{M}_i$ always encrypted to ciphertext in $\mathcal{M}_i$
- Rarely the case for random 1:1 functions $f: \mathcal{M} \to \mathcal{M}$, especially for large k



Simple Solution: Practical Security

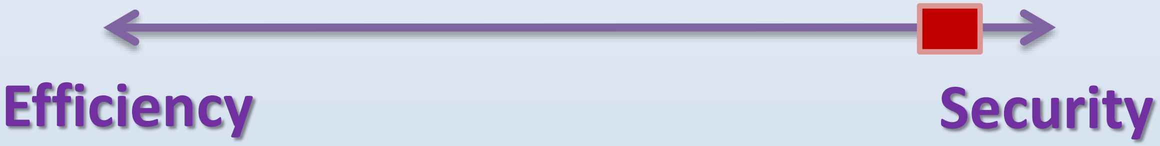
Simple solution:

- Divide $\mathcal{M}$ to subsets $\mathcal{M}_1, \ldots, \mathcal{M}_k$
- $\mathcal{M}_i$ defined by $\ell_i \Sigma_1^i, \ldots, \Sigma_\ell^i$
- Rank and encryption of $m \in \mathcal{M}_\mathrm{i}$ computed in relation to $\mathcal{M}_i$

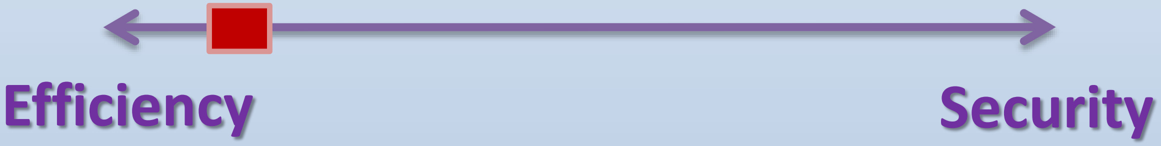
- $\mathcal{M}$ = names format:
  - 2-4 words
  - Every word upper-case followed by 1-10 lower-case
- $\mathcal{M}_i$ defines number of words + number of letters in each word
- "John Smith" can encrypt to "Angm Ojkri" but not to "Bar Refaeli"
- If only one of them is possible, adversary knows plaintext for sure

Optimizing Security-Efficiency Tradeoff

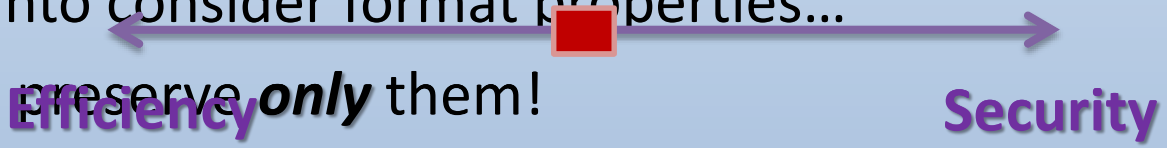
- Cycle walking inefficient since ignores format properties



- Simple solution insecure since preserves "cosmetic" message properties



- Want a "balanced" encryption scheme
  - Take into consider format properties…
  - …and preserve ***only*** them!



  - Need:
    - Framework of representing general formats
    - Method of ranking general formats

Representing General Formats: Framework

- Define building-blocks and operations
- Building blocks are called “primitives”
  - SSNs
  - CCNs
  - Dates (between minDate and maxDate)
  - Fixed-length strings with index-specific character-sets

(the format we saw before)

- Usually represent “rigid” formats
  - e.g., fixed length
- Can also represent “less rigid” formats
  - Variable-length strings over some alphabet

Representing General Formats: Framework (2)

- Define building-blocks and operations
- Operations allow constructing compound (and complex) formats from primitives
  - Operations preserve the parsing property: compound format can parse string to ingredients
- Compound formats are called “fields”
- Can construct format $\mathcal{F}$ from “smaller” formats $\mathcal{F}_1, \ldots, \mathcal{F}_k$ by:
  - Union
  - Concatenation:
    - $\mathcal{F} = \mathcal{F}_1 \cdot d_1 \cdot \mathcal{F}_2 \cdot \ldots \cdot d_{n-1} \cdot \mathcal{F}_n$, $d_1, \ldots, d_{n-1}$ are delimiter characters
    - $\mathcal{F} = \mathcal{F}_1 \cdot \ldots \cdot \mathcal{F}_k$ in some cases
  - Range: $\mathcal{F} = (\mathcal{F}_1 \cdot d)^k, min \leq k \leq max$

Constructing Compound Formats: Example

- $\mathcal{F}_1 =$ {A,B,…,Z}
- $\mathcal{F}_2 =$ length-k strings of lower-case letters, $1 \leq k \leq 10$
- $\mathcal{F}_3 =$SSNs
- Concatenation:
  - $\mathcal{F}_{word} = \mathcal{F}_1 \cdot \mathcal{F}_2$ gives words
  - $\mathcal{F} = \mathcal{F}_2 \cdot - \cdot \mathcal{F}_2$, e.g., “abc-def” or “aaaaa-bb”
- Union: $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_3$, e.g., “111223333” or “A”
- Range: $\mathcal{F}_{name} = (\mathcal{F}_{word} \cdot space)^k$ for $2 \leq k \leq 4$ gives names, e.g. “Bar Refaeli ” or “Louisa May Alcott ”

Ranking General Formats

- Define ranking for building-blocks
- Define ranking for operations
- Automatically gives ranking for compound formats:
  - Parse string to ingredients
  - Delegate ranking of substrings to ingredients
  - Use ranking for operations to “glue” ranks together

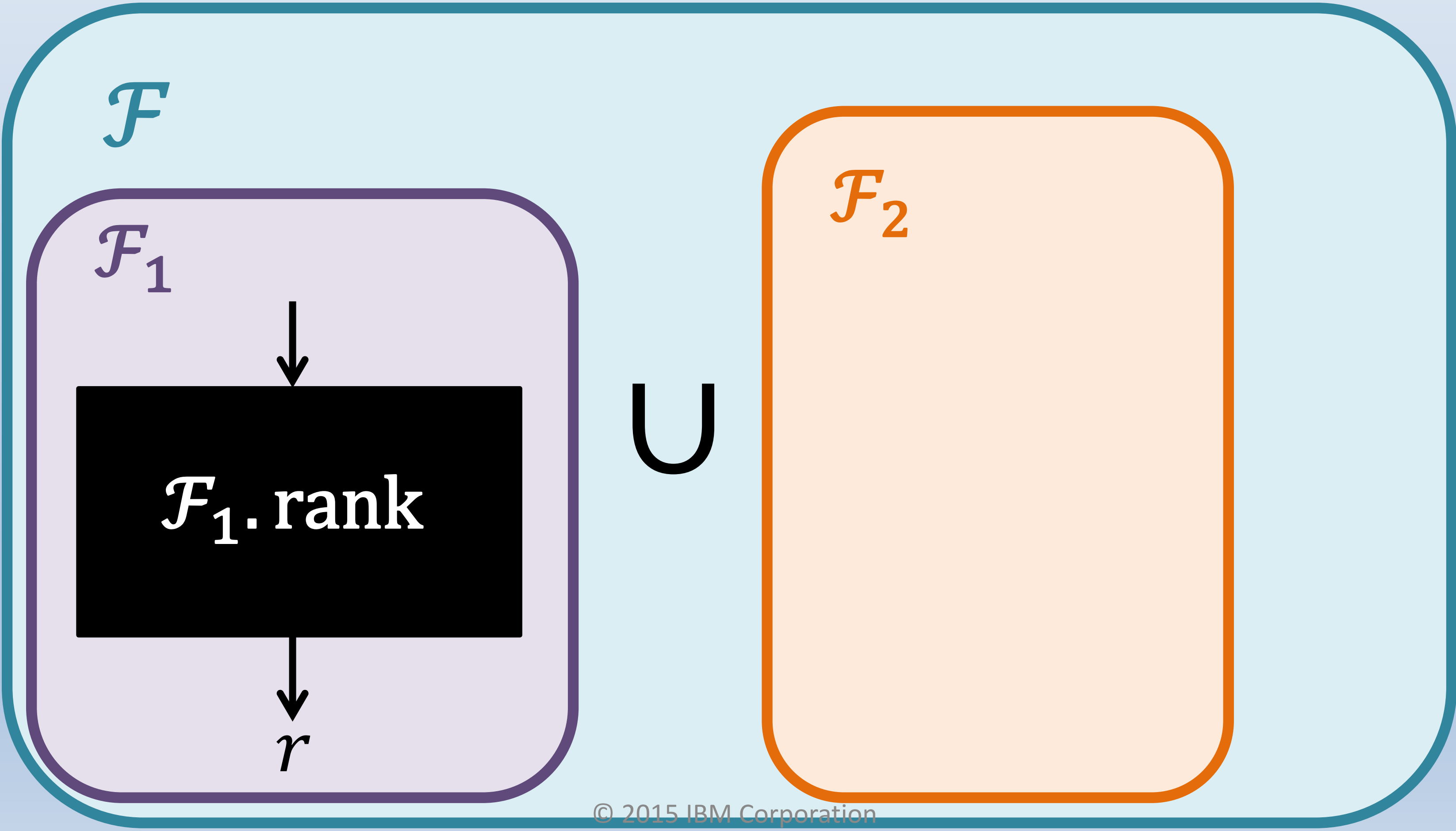
Ranking Primitives

- Ranking usually fairly simple:
  - SSNs: “basically” 9-digit numbers, remove illegal-SSNs smaller that given SSN
  - CCN: first 15 digits are the rank
  - Dates: count seconds since minDate
  - Fixed-length strings: Sum-and-Scale
  - Variable-length strings: Sum-and-Scale with same-length strings + offset by number of shorter strings
- Unranking more complex

lexicographic order!

Ranking Operations: Union

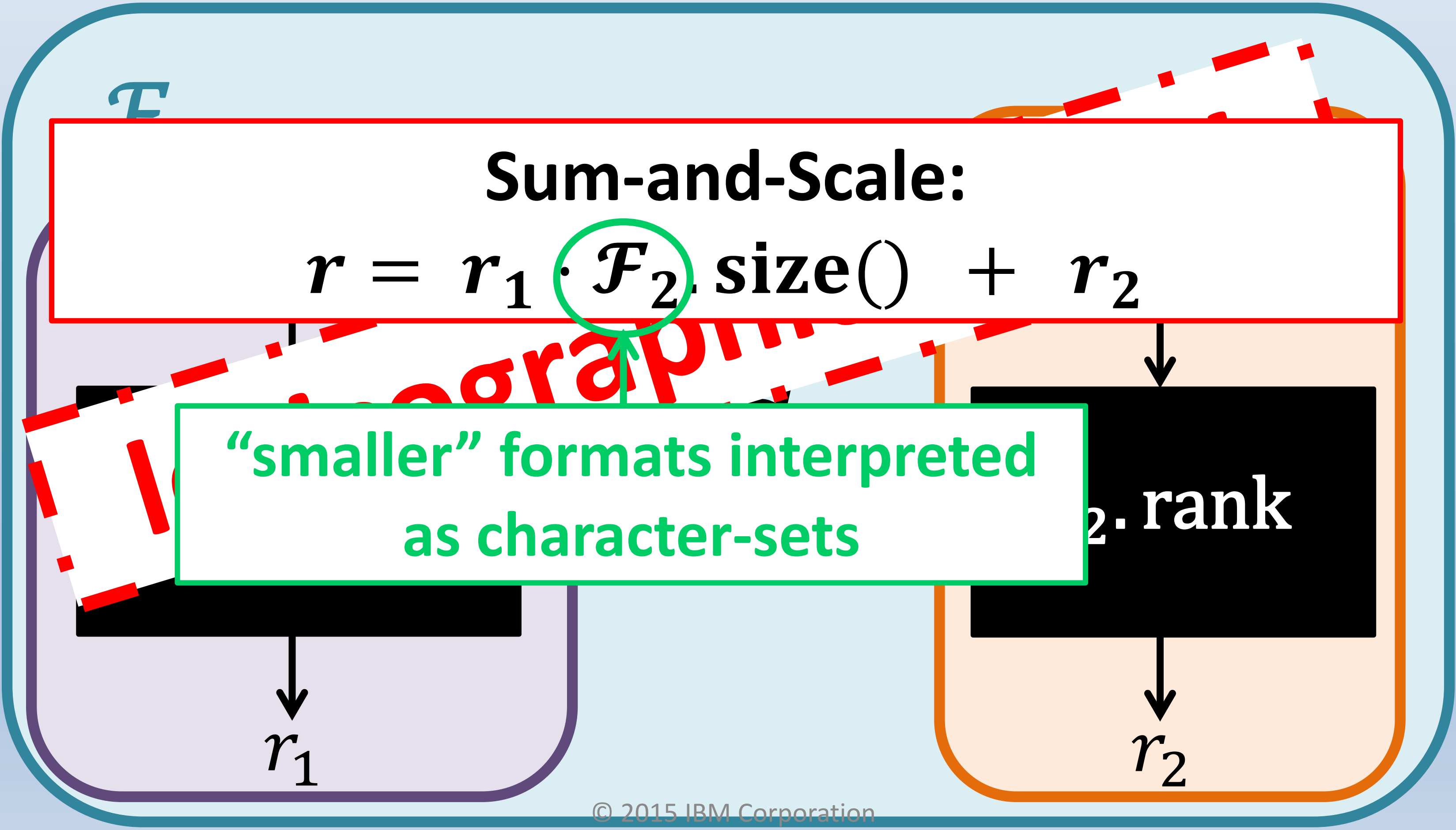
$$\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$$



Ranking Operations: Concatenation

$$\mathcal{F} = \mathcal{F}_1 \cdot d \cdot \mathcal{F}_2$$

$$m = m_1 \cdot d \cdot m_2$$



Sum-and-Scale:

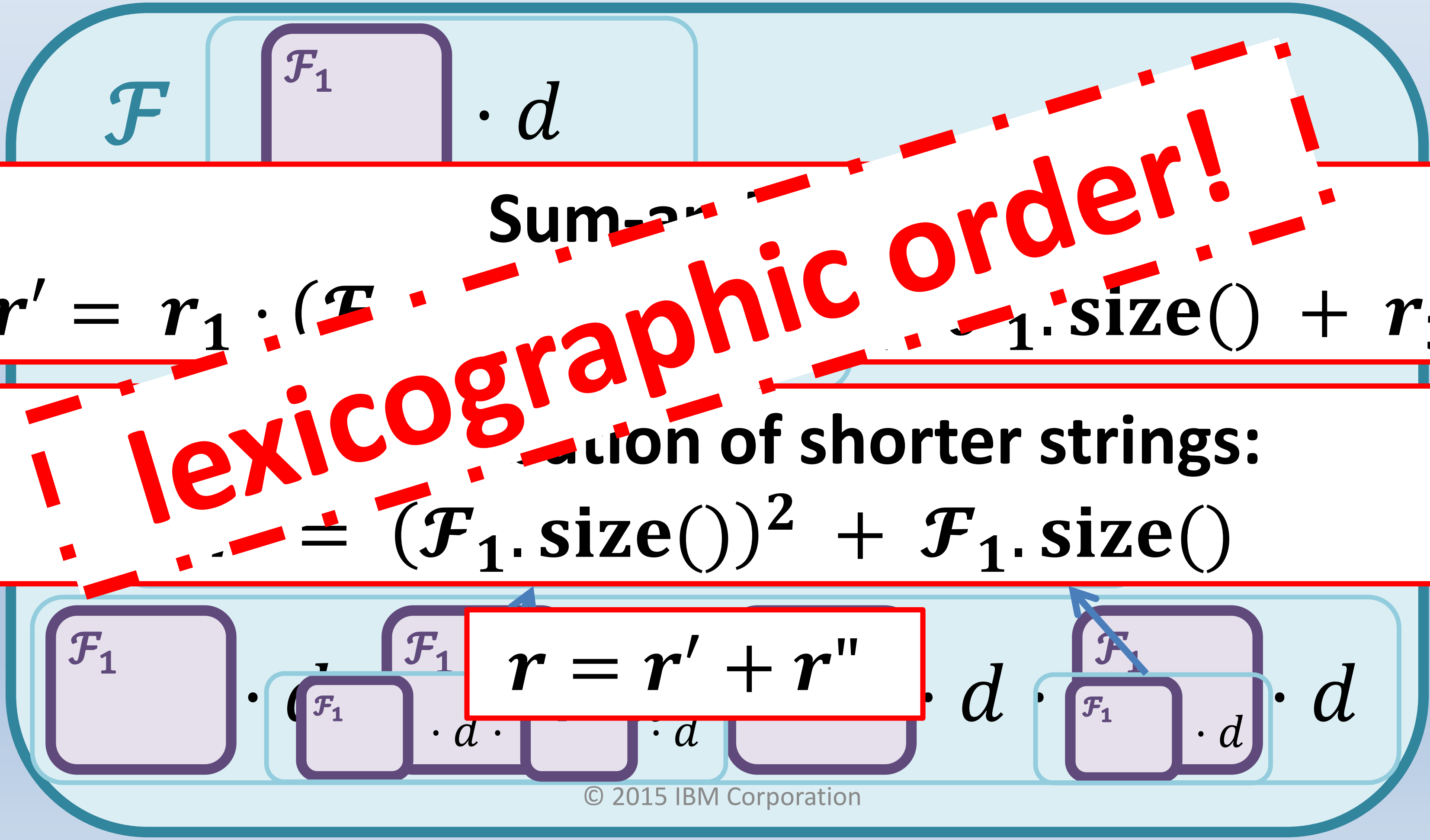
$$r = r_1 \cdot \mathcal{F}_2.\text{size}() + r_2$$

“smaller” formats interpreted as character-sets

Ranking Operations: Range

$$\mathcal{F} = (\mathcal{F}_1 \cdot d)^k,\ 1 \le k \le 4$$

$$m = m_1 \cdot d \cdot m_2 \cdot d \cdot m_3 \cdot d$$



Sum-a

$r' = r_1 \cdot (\mathcal{F}$ … $_1.\mathbf{size}() + r_3$

…ation of shorter strings:

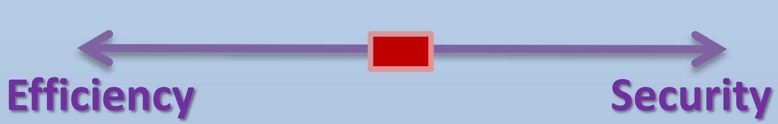
… $= (\mathcal{F}_1.\mathbf{size}())^2 + \mathcal{F}_1.\mathbf{size}()$

$r = r' + r''$

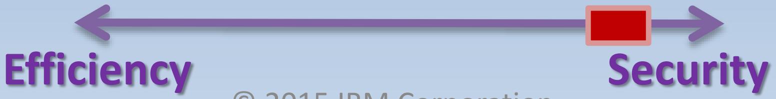
lexicographic order!

Our FPE: Analysis

- Security:
  - Only format properties preserved ⇒ security reduces to security of integer-FPE
  - Best security guarantee possible!
- Efficiency:
  - Ranking and unranking unavoidable in the Rank-then-Encipher method
  - Efficiency reduces to efficiency of integer-FPE
  - ✓ Medium-sized domains:

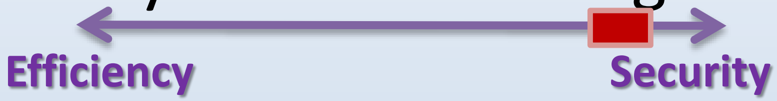


  - ✗ Large domains: only provably secure scheme [Bellare et al. ‘09] for range $\{0,1,\dots,M-1\}$ first factors M



Improving Efficiency For Large Formats

- Efficiency-security tradeoff for large formats:



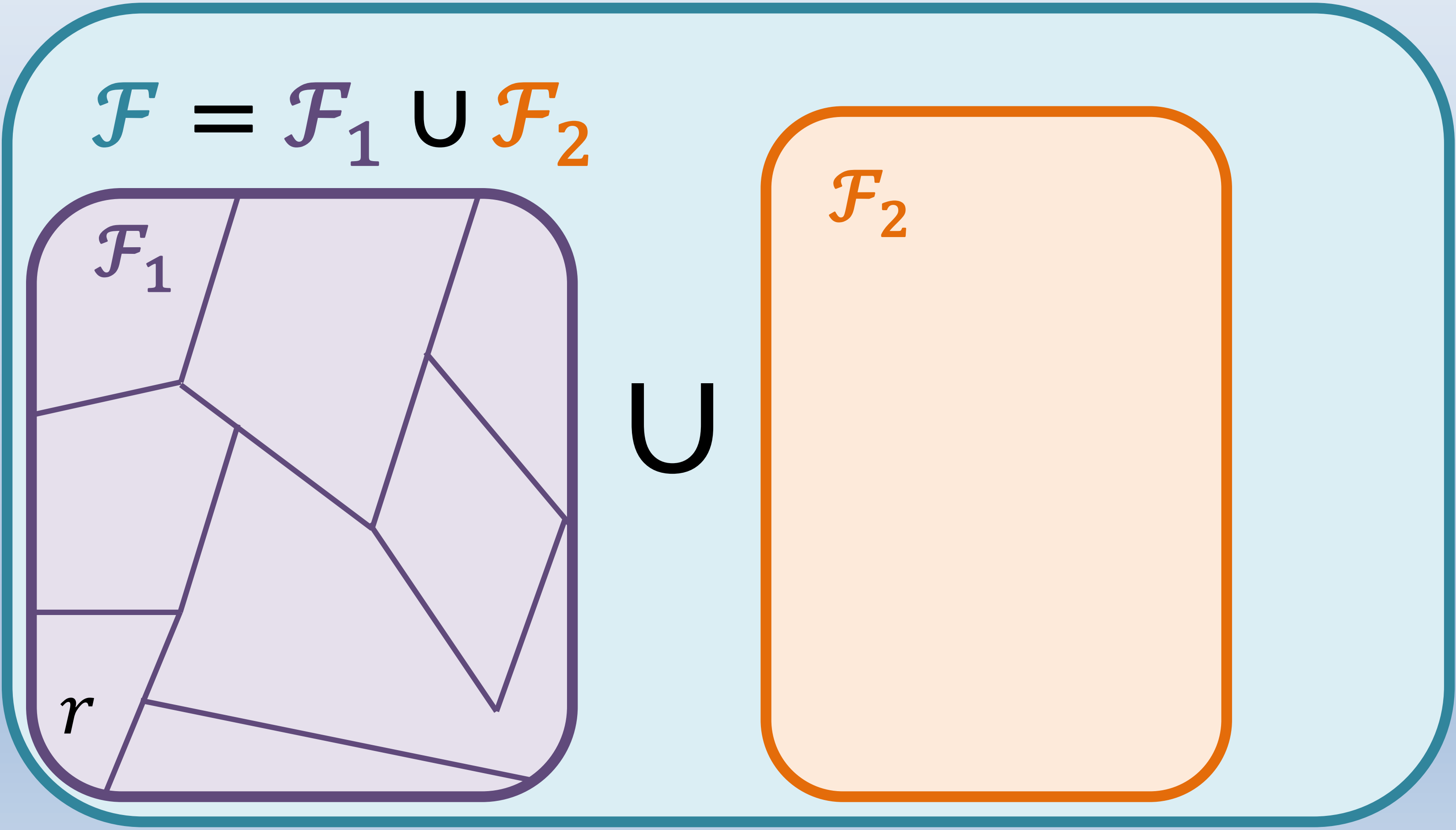
- 1st solution: use FFX for integer FPE
  - Has no rigorous security analysis
- 2nd solution: keep formats small ⇒ reduce format size
  - As we will see, this compromises security
  - We try to compromise as little as possible
- Partition message-space $\mathcal{M}$: $\mathcal{M} = \mathcal{M}_1 \cup \cdots \cup \mathcal{M}_n$
- But try to "hide" ***message-specific*** properties when possible
- Intuitively, try to increase the $\mathcal{M}_i$'s
  - Knowing $m \in \mathcal{M}_i$ still leaves "many unknowns"

The “Large Formats” Problem: Closer Look

- Inefficiency due to integer-FPE factoring domain size M
- Need to restrict domain size when calling integer-FPE
- Ranking and unranking is calculated in relation to M
- How do we rank in large formats?
- Our solution combines:
  - Delegating to sub-formats
  - Parsing message to substrings $m = m_1 \ldots m_n$ and applying Rank-the-Encipher **separately** to every m_i
- **Main challenge:** parsing m while hiding message-specific properties
  - Obtained by keeping sub-formats as large as possible

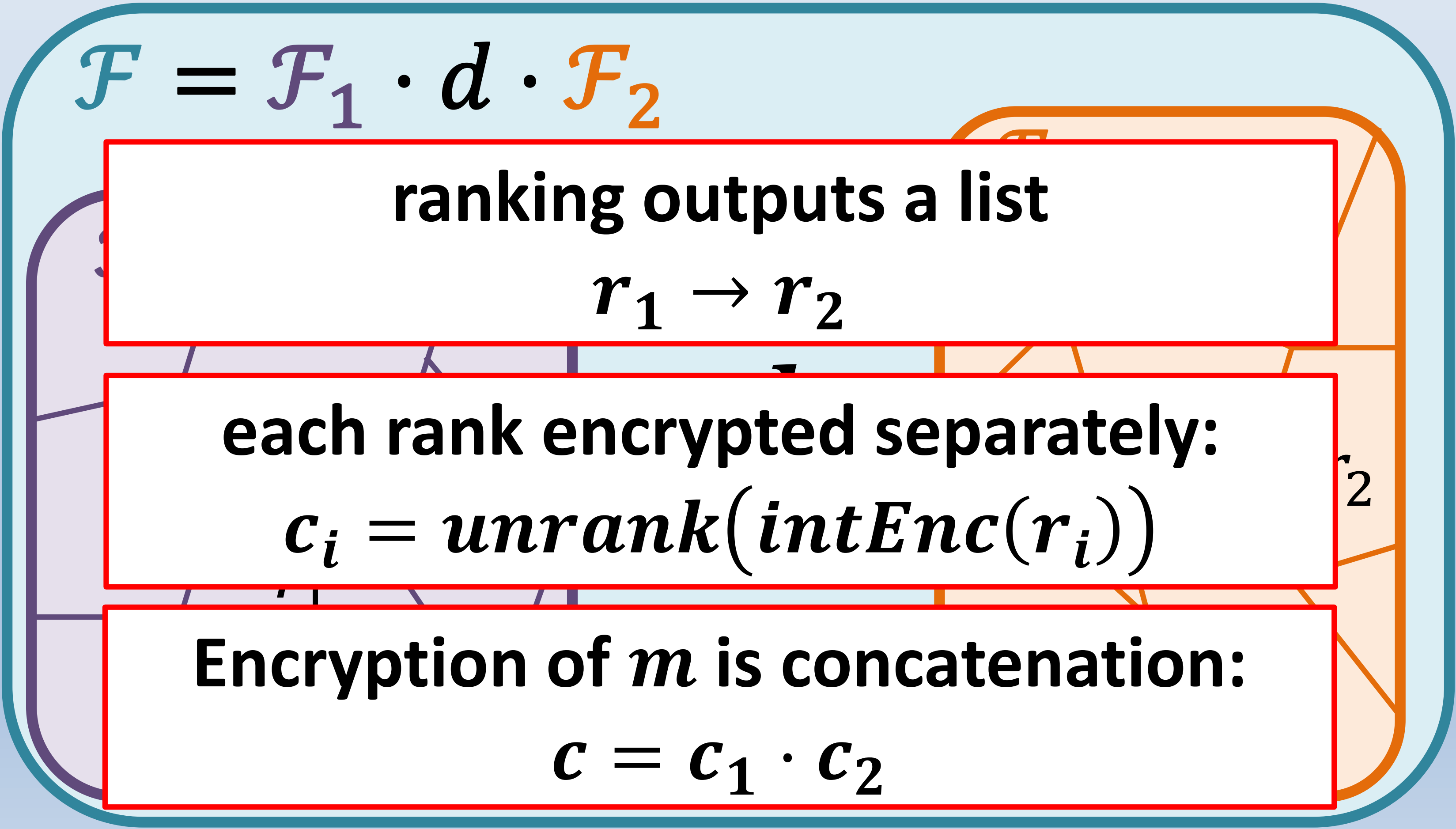
Parsing and Ranking Union

m



Parsing and Ranking Concatenation (1)

$$m = m_1 \cdot d \cdot m_2$$



$$\mathcal{F} = \mathcal{F}_1 \cdot d \cdot \mathcal{F}_2$$

ranking outputs a list

$$\boldsymbol{r_1 \to r_2}$$

each rank encrypted separately:

$$\boldsymbol{c_i = unrank\big(intEnc(r_i)\big)}$$

Encryption of $\boldsymbol{m}$ is concatenation:

$$\boldsymbol{c = c_1 \cdot c_2}$$

Parsing and Ranking Concatenation (2)

$$m = m_1 \cdot d_1 \cdot m_2 \cdot d_2 \cdot m_3 \cdot d_3 \cdot m_4 \cdot d_4 \cdot \boldsymbol{m_5}$$

$\mathcal{F}$ = $\cdot\ \mathcal{F}_5$

ranking outputs a list

$$\boldsymbol{r' \rightarrow r'' \rightarrow r'''}$$

$\mathcal{F}_1$ r_1

each rank encrypted separately:

$$\boldsymbol{c' = unrank\big(intEnc(r')\big)}$$

$$\boldsymbol{c'' = unrank\big(intEnc(c'')\big)}$$

$$\boldsymbol{c''' = unrank(intEnc(r'''))}$$

$\mathcal{F}_5$ r_5

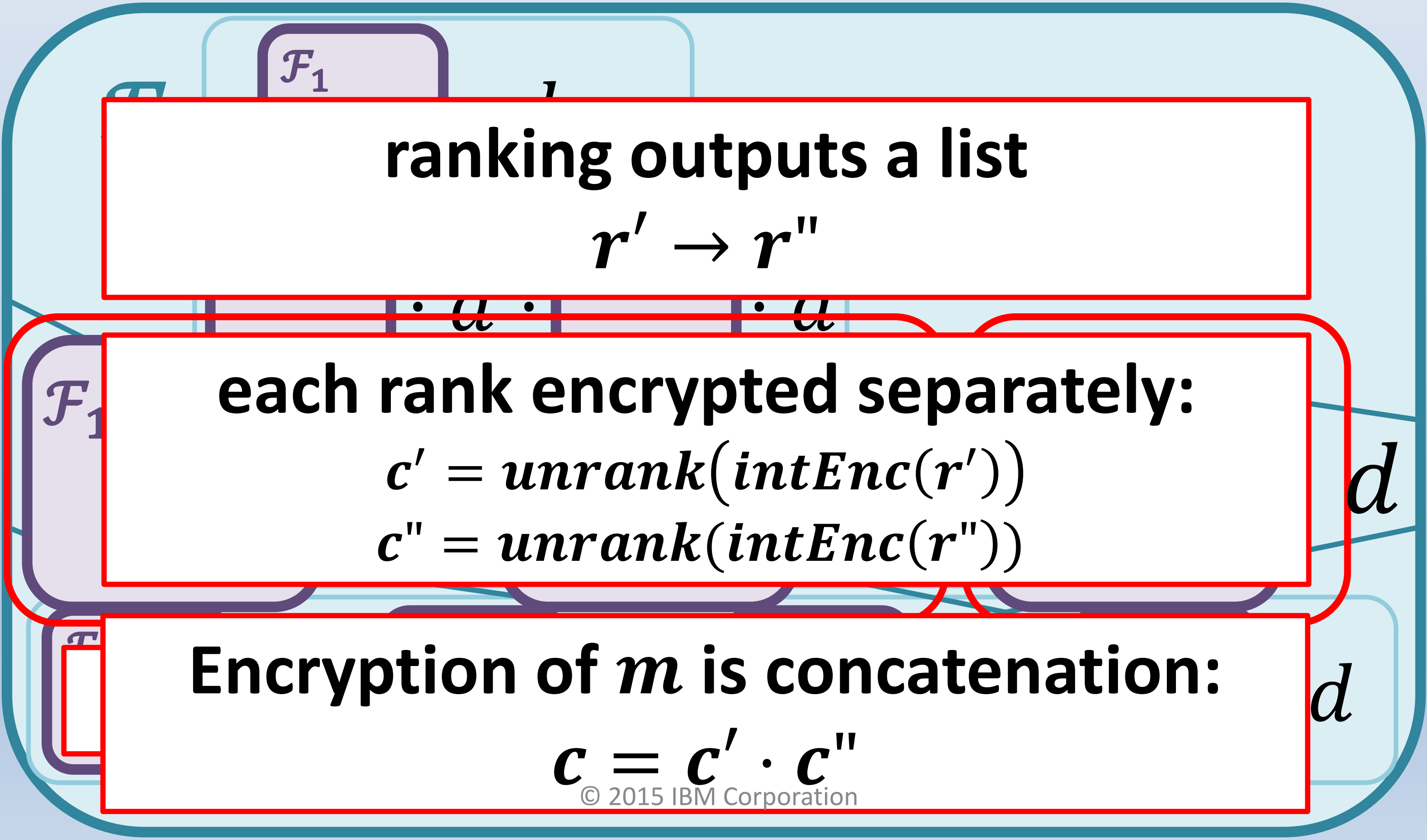
$\boldsymbol{r'} =$ $= \boldsymbol{r_5}$

Encryption of $\boldsymbol{m}$ is concatenation:

$$\boldsymbol{c = c' \cdot c'' \cdot c'''}$$

Parsing and Ranking Range

$$\mathcal{F} = (\mathcal{F}_1 \cdot d)^k,\ 1 \leq k \leq 4$$
$$m = \boldsymbol{m_1} \cdot d \cdot \boldsymbol{m_2} \cdot d \cdot \boldsymbol{m_3} \cdot d$$



ranking outputs a list

$$\boldsymbol{r'} \rightarrow \boldsymbol{r''}$$

each rank encrypted separately:

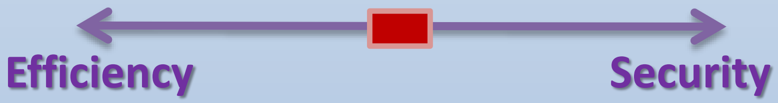
$$\boldsymbol{c'} = \boldsymbol{unrank}(\boldsymbol{intEnc}(\boldsymbol{r'}))$$
$$\boldsymbol{c''} = \boldsymbol{unrank}(\boldsymbol{intEnc}(\boldsymbol{r''}))$$

Encryption of $\boldsymbol{m}$ is concatenation:

$$\boldsymbol{c} = \boldsymbol{c'} \cdot \boldsymbol{c''}$$

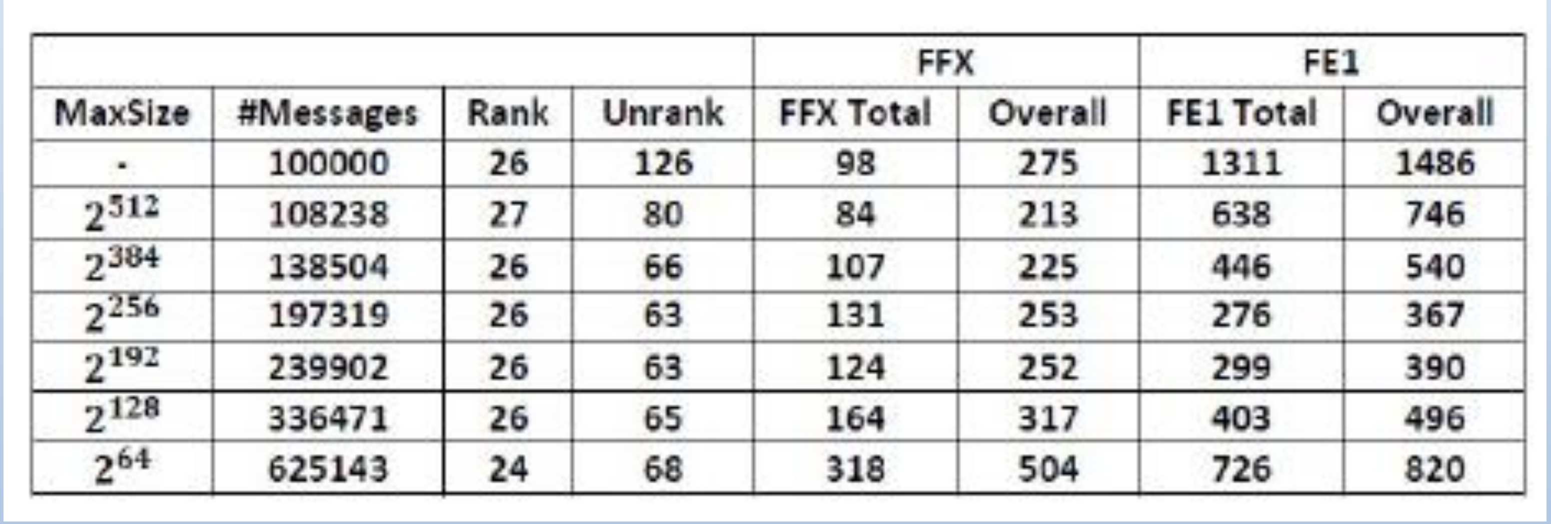
Security Of Our FPE

- Format sub-dividing preserve *some* message-specific properties
- The larger the sub-format, the smaller the probability of reversing encryption
- Choosing parameters “correctly” ⇒ “reasonable” tradeoff



Our FPE: Evaluation

- Federal Election Commission (FEC) reports:
  - Name, home address, employer, job title
    - Format size ~ $\mathbf{2^{856}}$



				FFX		FE1	
MaxSize	#Messages	Rank	Unrank	FFX Total	Overall	FE1 Total	Overall
-	100000	26	126	98	275	1311	1486
2^{512}	108238	27	80	84	213	638	746
2^{384}	138504	26	66	107	225	446	540
2^{256}	197319	26	63	131	253	276	367
2^{192}	239902	26	63	124	252	299	390
2^{128}	336471	26	65	164	317	403	496
2^{64}	625143	24	68	318	504	726	820

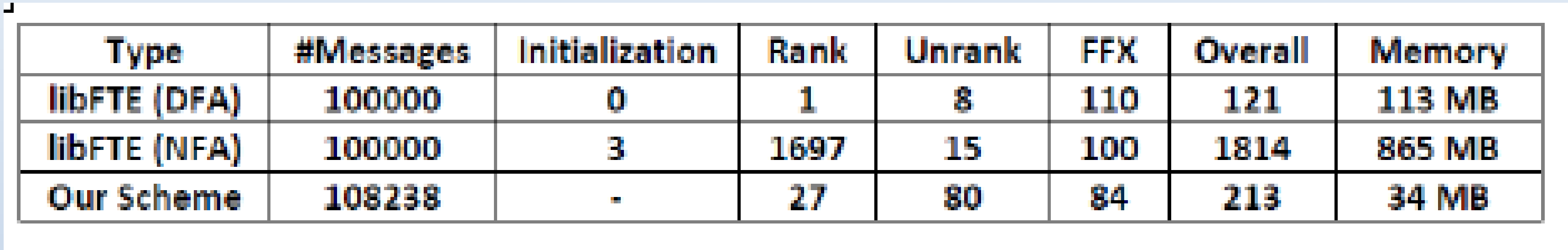
  - FFX achieves better performance
  - Splitting significantly improves the FE1 running time
    - Setting maxSize < $\mathbf{2^{256}}$ has no efficiaency gain

Concurrent Work

- libFTE [Luchaup et al. ’14]
  - Also employ RtE
  - Format represented by *regexp*
    - Regexp->DFA/NFA
    - Rank/Unrank using DFA/NFA
- Limitations:
  - Designed for developers:
    - Defining new format (regexp) requires a developer’s involvement
    - outputs several possible schemes out of which developer choses the most appropriate one
    - resultant scheme could have poor performance and there is no way to know whether a different regex would give better performance

Concurrent Work (Cont.)

- Performance of our scheme compared to libFTE:



Type	#Messages	Initialization	Rank	Unrank	FFX	Overall	Memory
libFTE (DFA)	100000	0	1	8	110	121	113 MB
libFTE (NFA)	100000	3	1697	15	100	1814	865 MB
Our Scheme	108238	-	27	80	84	213	34 MB

- Running Time: libFTE is ~ twice as fast as our approach
- Memory Usage: libFTE uses ~ 3 time more memory

Our FPE: Practical Summary

- We provide an FPE for **general** formats
  - First framework for efficiently representing general formats
  - First scheme to eliminate cycle-walking
    - Efficiency can be measured!
  - Optimal security guarantee
  - Support of large formats
    - With best security guarantee under size limitation
- Ingredients:
  - Framework for defining general formats
  - Efficient ranking and unranking methods for general formats
  - Support of large format
    - Through user-defined upper-bound on permissible format sizes

Thanks For Listening!