Practical Solutions for Format-Preserving Encryption

Mor Weiss

Joint work with Boris Rozenberg and Muhammad Barham

Research conducted while all authors were at IBM Research Labs, Haifa
Why Format Preserving Encryption?
Why Format Preserving Encryption?
Why Format Preserving Encryption?

Problem (1): encrypted entry incompatible with database entry structure

Non-solution (1): generate new tables
Why Format Preserving Encryption?
Why Format Preserving Encryption?
Why Format Preserving Encryption?

Problem (2): encrypted entry incompatible with applications using data

Non-solution (2): re-write applications
Talk Outline

• Definitions
• Methodology for format-preserving encryption of general formats
• Analysis of known constructions
• GFPE
• Optimizations for large formats
Format-Preserving Encryption: Definition

• A deterministic private-key Encryption Scheme $\Pi$:
  – Message space $\mathcal{M}$
  – Randomized $\text{KeyGen}: \mathbb{N} \rightarrow \mathcal{K}$
  – Deterministic $\text{Enc}: \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$
  – Deterministic $\text{Dec}: \mathcal{K} \times \mathcal{C} \rightarrow \mathcal{M}$

• Notation: $\text{Enc}_k = \text{Enc}(k, \cdot)$, $\text{Dec}_k = \text{Dec}(k, \cdot)$

• Encryption key random and secret ⇒ encryption “hides” plaintext

• Standard encryption: ciphertexts usually “look like garbage”, possibly causing
  – Applications using data to crash
  – Tables designed to store data unsuitable for storing encrypted data

• ⇒ Sometimes plaintext properties should be preserved

• Format-Preserving Encryption (FPE): $\mathcal{M} = \mathcal{C}$
  – $\text{Enc}_k$ is a permutation over plaintext space $\mathcal{M}$
  – Ciphertexts have same format as plaintexts!
FPE: Definition (cont.)

- **Correctness:** for every $k \in \mathcal{K}$ and every $m \in \mathcal{M}$
  \[
  \text{Dec}_k(\text{Enc}_k(m)) = m
  \]

- **Secrecy:**
  - For *secret and random* $k \in \mathcal{K}$
  - Hierarchy of security notions [BRRS`09]
  - **Strongest:** random $k \Rightarrow \text{Enc}_k$ close to pseudorandom permutation
    - An “overkill” for many typical applications
      - Guaranteed security against (improbable) attacks incurs expensive overhead
  - **Weakest:** Message Recovery
    - Only require that adversary cannot *completely* recover message
      - Even given advantageous distribution over $\mathcal{M}$
    - Very weak: adversary may learn some message properties
What We Know About FPE

- Term coined by Terence Spies, Voltage Security’s CTO
- First formal definitions due to [BRRS’09]
- Constructions for specific formats
  - Social Security Numbers (SSNs) [Hoo’11]
  - Credit Card Numbers (CCNs)
  - Dates [LJLC’10]
  - …
- **Drawbacks:**
  - Designed for specific formats (different scheme for every format)
  - New encryption techniques, little (if any) security analysis
- **Integral domains** \{1, ..., M\} [BR’02,BRRS’09]
- “Almost integral” domains \(\mathcal{M} = \{1, ..., m\}^n\) for \(n, m \in \mathbb{N}\)
  - Methods described as early as 1981
  - FFX [BRS’10], BPS [BPS’10] submitted to NIST for consideration
Format-Preserving Encryption for General (Complex) Formats
Techniques for General-Format FPE (Part 2)

- **Rank-then-Encipher (RtE) [BRRS`09]:** general-format FPEs from int-FPE
  - Order $\mathcal{M}$ arbitrarily: $\text{rank}: \mathcal{M} \rightarrow \{1, \ldots, M\}$
Techniques for General-Format FPE (Part 2)

- Rank-then-Encipher (RtE) [BRRS`09]: general-format FPEs from int-FPE
  - Order $\mathcal{M}$ arbitrarily: \textbf{rank}: $\mathcal{M} \rightarrow \{1, \ldots, M\}$
• **Rank-then-Encipher (RtE) [BRRS`09]:** general-format FPEs from int-FPE
  
  – Order $\mathcal{M}$ arbitrarily: $\text{rank}: \mathcal{M} \rightarrow \{1, \ldots, M\}$
  
  – To encrypt message $m$:
    
    • **Rank $m$:** $i = \text{rank}(m)$
    • **Encipher $i$:** $j = \text{intE}(K, i)$
    • **Unrank $j$:** $c = \text{rank}^{-1}(j)$
Techniques for General-Format FPE (Part 2)

- **Rank-then-Encipher (RtE) [BRRS`09]:** general-format FPEs from int-FPE
  
  - Order $\mathcal{M}$ arbitrarily: $\text{rank}: \mathcal{M} \rightarrow \{1, \ldots, M\}$
  
  - To encrypt message $m$:
    - **Rank $m$:** $i = \text{rank}(m)$
    - **Encipher $i$:** $j = \text{intE}(K, i)$
    - **Unrank $j$:** $c = \text{rank}^{-1}(j)$
Techniques for General-Format FPE

- **Rank-then-Encipher (RtE) [BRRS`09]:** general-format FPE from integer-FPE
  - Order $\mathcal{M}$ arbitrarily: $\text{rank}: \mathcal{M} \rightarrow \{1, \ldots, M\}$
  - To encrypt plaintext $m$:
    - Rank $m$: $i = \text{rank}(m)$
    - Encipher $i$: $j = \text{integerEnc}_k(i)$
    - Unrank $j$: $c = \text{rank}^{-1}(j)$

- **Security:** from security of integer-FPE
  - rank not meant to, and does not, add security

- **Efficiency:** only if rank, unrank are efficient

- **Main challenge (1):** design efficient rank procedure
  - “Meta” ranking technique for regular languages [BRRS`09]

- **Main challenge (2):** representing formats
FPEs for General Formats:

Previous solutions
Simplification-Based FPE [MYHC`11,MSP`11]

- Represent formats as union of simpler sub-formats
  - Plaintexts interpreted as strings
  - $\mathcal{M}$ divided into subsets $\mathcal{M}_1, \ldots, \mathcal{M}_k$ defined by
    - Length
    - Index-specific character sets

- Encrypt each $\mathcal{M}_i$ *separately* using Rank-then-Encipher
  - Ranking computed using generalized lexicographic ordering

$F_{name}$: format of valid names
**Name:** 1-4 space-separated words
**Word:** upper case letter followed by 1-15 lower case letters

**Subsets:**
- $\mathcal{M}_1$ contains Al
- $\mathcal{M}_2$ contains Tal
- ...
- $\mathcal{M}_{15}$ contains Muthuramakrishna
- $\mathcal{M}_{16}$ contains El Al
Simplification-Based FPE: Security Concerns

- **The problem:** encryption preserves *plaintext-specific* properties
  - **Reason:** each sub-format $\mathcal{M}_i$ encrypted separately
  - “John Doe” can encrypt “Jane Roe” but not “Johnnie Dee”
  - If only one of them is possible, adversary knows plaintext for sure

- Simplification-based FPE is Message-Recovery insecure \cite{WRB`15}
  - MR (message recovery) is the weakest notion
  - Implies insecurity according to other FPE security notions

- **Reason:** ciphertext *length* reveals plaintext length, can be used to recover message
Simplification-Based FPE: Experimental Results

- Our experiments performed on 1M records of the Federal Election Commission (FEC) reports of 2008-2012
  - Regulates campaign finance legislation in the US
  - Report lists all donors over $200:
    - Name
    - Town
    - Employer
    - Job title

- Attack model reflects typical threat
  - Data stored at remote server
  - Attacker has access to all or part of database
  - No access to secret encryption key
  - $A$ may have prior knowledge
Simplification-Based FPE: Experimental Results (Cont.)

When $\mathcal{A}$ recovers *only name* column

- If we’re lucky – Bar in 7% of donors whose encryptions match only 100 entries
Simplification-Based FPE: Experimental Results (Cont.)

When $A$ recovers **name and town** columns

- If we’re lucky, Bar in 7% of donors whose encryptions match only 2 entries
- **Pretty likely** that Bar in 44% of donors whose encryptions match only 100 entries
Simplification-Based FPE: Experimental Results (Cont.)

When $\mathcal{A}$ recovers the entire database

- For all donors: encryptions match $\leq 250$ entries!
- Most likely Bar in 71% of donors whose encryption matches only 2 entries!
GFPE [WRB`15]  
FPE “Wish List”

• **Functionality, efficiency:**
  – *Simple* method of representing formats
  – *Efficient* rank, unrank procedures

• **Security:** preserve *only format-specific* properties
  – Hide *all plaintext-specific* properties

**The Scheme:**

• Encryption\decryption using Rank-then-Encipher
  – Support integer-FPEs for integral *and* almost integral domains

• **Main challenge:** user-friendly format representation
  – Scheme is user-oriented

• **Structure:** formats represented using bottom-up framework
  – *“Basic” building-blocks (primitives)*
    • Usually “rigid” formats (e.g., SSNs, CCNs, dates, fixed-length strings…)
    • Also “less rigid” formats (e.g., variable-length strings)
  – **Operations** used to construct complex formats
GFPE: Representing Formats

- **“Basic” building-blocks (primitives):**
  - $\mathcal{F}_{upper} = \{A,B,...,Z\}$
  - $\mathcal{F}_{lower} = \text{length-}k \text{ lower-case letter strings, } 1 \leq k \leq 15$
  - $\mathcal{F}_{ssn} = \text{social-security numbers (SSNs)}$

- **Operations:**
  - **Concatenation:**
    - $\mathcal{F} = \mathcal{F}_1 \cdot ... \cdot \mathcal{F}_k$
    - Words: $\mathcal{F}_{\text{word}} = \mathcal{F}_{\text{upper}} \cdot \mathcal{F}_{\text{lower}}$
    - $\mathcal{F} = \mathcal{F}_1 \cdot d_1 \cdot \mathcal{F}_2 \cdot ... \cdot d_{n-1} \cdot \mathcal{F}_n$ ($d_1, ..., d_{n-1}$ are delimiters)
  - **Range:** $\mathcal{F} = (\mathcal{F}_1 \cdot d)^k, \min \leq k \leq \max$
    - Names: $\mathcal{F}_{\text{name}} = (\mathcal{F}_{\text{word}} \cdot \text{space})^k$ for $1 \leq k \leq 4$
  - **Union:** $\mathcal{F} = \mathcal{F}_1 \cup ... \cup \mathcal{F}_k$
    - “Names or SSNs”: $\mathcal{F} = \mathcal{F}_{\text{name}} \cup \mathcal{F}_{\text{ssn}}$
Example: Representing Addresses

name    house #     street    city    zip

• $F_{name} = (F_{word} \cdot space)^k$ for $1 \leq k \leq 4$ (range)
• $F_{num} = \{1, \ldots, 100\}$ (integral domain)
• $F_{zip} = \{0, 1, \ldots, 9\}^5$ (fixed length string)
• Valid addresses obtained through concatenation:
  $F_{add} = F_{name} \cdot F_{num} \cdot F_{name} \cdot F_{name} \cdot F_{zip}$
  name    house #    street    city    zip
GFPE: Encryption

• Use Rank-then-Encipher method
  – Use “off-the-shelf” integer-FPE schemes
  – Inherit security of underlying integer-FPE

• **Challenge**: how to rank and unrank?

• Define ranking for **primitives** and **operations**

• Rank of compound formats computed top-down:
  – Parse string to components
  – Delegate substring ranking to format components
  – “Glue” ranks together using ranking for operations
Example: Ranking Concatenation

\[ F = F_1 \cdot d \cdot F_2 \]

\[ m = m_1 \cdot d \cdot m_2 \]
Example: Ranking Concatenation

\[ F = F_1 \cdot d \cdot F_2 \]

\[ m = m_1 \cdot d \cdot m_2 \]

\[ r = r_1 + r_2 \cdot F_1 \cdot \text{size()} \]

Scale by size of sub-formats
GFPE: Supporting Large Formats

• Scheme supports integer-FPEs [BR`02,BRRS`09]
  – Only provably secure schemes

• Integer-FPEs are inefficient for large domains!
  – Require factoring domain size

• Supporting large formats: keep formats small
  – Divide large formats, encrypt each sub-format separately
  – Minimize security loss by “hiding” plaintext-specific properties:
    • Division according to format structure
    • Maximizing sub-format size
  – maxSize determined by user-defined performance constraints
Example: Dividing Address Format

name  house #  street  city  zip

• Valid addresses obtained through concatenation:

\[ F_{\text{add}} = (F_{\text{name}} \cdot F_{\text{num}}) \cdot (F_{\text{name}} \cdot F_{\text{street}}) \cdot (F_{\text{name}} \cdot F_{\text{zip}}) \]

• Jane Doe 23  Delaford  New York 12345
• Jane Doe 23  Bedford  New York 90210
• Smaller maxSize ⇒ further division
  – E.g., \( F_{\text{name}} \) divided according to number of words in name
Security of GFPE: Large Formats

• Format division introduces complications in ranking and unranking
  – Generalize rank, unrank to lists of ranks

• GFPE format-division strategy:
  – Usually hides all plaintext-specific properties
  – Small maxSize ⇒ may preserve some properties in huge formats
    • But properties defined by “semantic” sub-format, not “cosmetic” plaintext properties
  – Maximizes sub-format size
    • Minimizes possibilities of attacks

• “Wise” choice of parameters ⇒ “reasonable” tradeoff
Security of GFPE: Large Formats (2)

• Given user-defined efficiency constraints, we can evaluate security loss

• **Experimental results:** compared GFPE with simplification-based FPE
  
  – On 1M records of the Federal Election Commission (FEC) reports of 2008-2012

• **Simplification-based FPE:** every encrypted record matches at most **250** records

• **GFPE:** when maximizing efficiency
  
  – 99% encrypted records match > **1000** records
  – 94% encrypted records match > **10,000** records
  – 67% encrypted records match > **100,000** records
  – ...
Concurrent Work: libFTE [LDJRS’14]

- Library for format-preserving and format transforming encryption of general formats
  - Also based on Rank-then-Encipher
    - Support less integer-FPE schemes
  - Formats represented using Regular Expressions
  - Ranking uses automatons (deterministic or non-deterministic)
- Different goal: developer-oriented
  - Defining new formats
  - Choosing “right” scheme to use
- Same security guarantee
- Comparable “best case” efficiency
  - libFTE “worst case” can be much worse
Summary

• **Goal:** FPE for general formats

• Analyze existing schemes
  – Show security vulnerabilities
  – Inefficiencies also exist

• Propose a new FPE scheme for general formats
  – Based on Rank-the-Encipher
  – Simple and efficient methodology of representing and ranking formats
  – **Flexible scheme:**
    • Can use any FPE for integral or almost integral domains
    • Easy to add new primitives: just provide rank, unrank
    • User-controlled efficiency-security tradeoff (through \( maxSize \) param)
THANK YOU