Cryptanalysis of a Fast Encryption Scheme for Databases

Stéphane Jacob

Project-team SECRET,
INRIA Paris-Rocquencourt

June 15, 2010

Outline

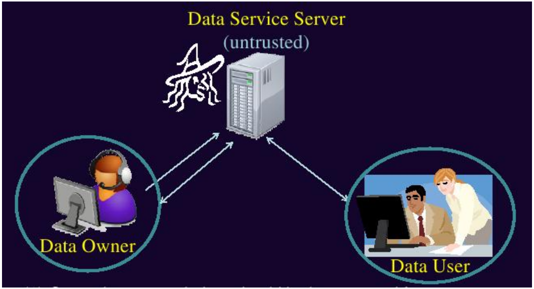
1. Database Encryption
   - Context
   - Description of the FCE Encryption Scheme

2. An Attack against FCE
   - Concept
   - Algorithm
   - Simulation Results and FCE Variant

Outline

1. Database Encryption
   - Context
   - Description of the FCE Encryption Scheme
2. An Attack against FCE
   - Concept
   - Algorithm
   - Simulation Results and FCE Variant

Data As Service: Outsourced Data



- The client is trusted but has low storage/computation capacities.
- The server is untrusted but has high storage/computation capacities.

Naïve but Impractical Examples

- Encrypt the whole database (*e.g.* AES in CBC mode):
  - every query requires a full database decryption.
- Encrypt every field separately (with its own IV):
  - every query requires a full column decryption,
  - it requires a lot of padding.

General Goal

We want to:

- Prevent information leaking.
- Detect data falsification.
- Use fast encryption and decryption algorithm.
- Keep a good structure in order to be able to query the database.

There is a trade-off between functionalities and security, and a perfect solution does not exist.

Some Methods Proposed by the Database Community

- Order preserving encryption (OPE).
- Prefix preserving encryption (PPE).

Outline

1. Database Encryption
   - Context
   - Description of the FCE Encryption Scheme

2. An Attack against FCE
   - Concept
   - Algorithm
   - Simulation Results and FCE Variant

Fast Comparison Encryption

T. Ge and S. Zdonik.
Fast, secure encryption for indexing in a column-oriented DBMS.
In *International Conference on Data Engineering - ICDE 2007*, pages 676–685. IEEE, 2007.

- This encryption scheme allows fast comparison *i.e.* it allows to quickly decide if 2 data are different.
- The comparison of 2 encrypted data with "Early Stopping":
  - starts from the most significant byte,
  - proceeds byte by byte,
  - stops once a difference is found.

This can be obtained by using stream ciphers.

Database Storage

"row-store"

id	name	health expenses
1	Dupont	0
2	Dupond	42000
...	...	...

$\longrightarrow$

"column-store"

id	name
1	Dupont
2	Dupond
...	...

id	health expenses
1	0
2	42000
...	...

Encryption with $(s_t, t \geq 0)_{\{K,\ \text{page number}\}}$

id	health expenses
1	$0 \oplus s_0$
2	$42000 \oplus s_1$
...	...

Encryption Algorithm: FCE

- There is a unique secret k-bit length key K for the whole database.
- Encryption proceeds page by page.
- To each plain text page corresponds a polynomial:

$$P(x) = ax^3 + bx^2 + cx + d \bmod p \text{ with } a, b, c, d \in [0, p-1],$$

where a, b, c, and d are computed from K and the page number j using a classic block encryption, *e.g.* AES(K, j).

Parameters:

- key size $k = 2^\kappa = 2^{15}$ bits,
- page size $p = 2^{\kappa+1} = 2^{16}$ bytes,
- a, b, c, d size: 64 bits per 64Kbytes page.

Encryption Algorithm: FCE

Page encryption:

- $(a, b, c, d) \leftarrow$ AES$(K, \text{page nb})$ with $a, b, c, d \in [0, 2^{\kappa+1} - 1]$
- $P(x) = ax^3 + bx^2 + cx + d \bmod 2^{\kappa+1}$

For i from 0 to $p - 1$:

- $d_i = P(i) \bmod k$
- $v_i = K_{\{d_i \to d_i+7\}}$ is the key byte starting at the bit d_i:
  $K_0, K_1, \ldots, \underbrace{K_{d_i}, \ldots, K_{d_i+7}}_{v_i = K_{\{d_i \to d_i+7\}}}, \ldots, K_{k-1}$
- $c_i = m_i \oplus v_i$

Outline

Known Plain Text Attack

Input: half a page of plain text (2^κ bytes m_i) and the corresponding half page of cipher text (2^κ bytes c_i).

Output: key K (and thus all the polynomials of the different pages).

Important Remark

$\forall\ d'$, the keys $\left\{ \begin{array}{l} K' = K \ggg d' \\ P'(x) = ax^3 + bx^2 + cx + (d - d') \end{array} \right.$ are equivalents.

Therefore, we are looking for $\widetilde{K} = K \ggg d$.

Naïve Attack

- For each triple (a, b, c), we try to rebuild the key from the keystream.
- In case of success, we search for d by computing the page polynomial, from the page number and the key we built.

Cost: 2^{48+15} polynomial evaluations + 2^{15} AES computations.

Main Idea

Searching for (α, β, γ) antecedents of $(1, 2, 3)$ by $\widetilde{P}$, *i.e.* triples (α, β, γ) where $\widetilde{P}(\alpha) = 1,\ \widetilde{P}(\beta) = 2,\ \widetilde{P}(\gamma) = 3$.
Indeed, v_0 (which is known), v_α, v_β and v_γ overlap:

$$v_0 = \widetilde{K}_0, \underbrace{\widetilde{K}_1, \dots, \widetilde{K}_7}$$

$$v_\alpha = \overbrace{\widetilde{K}_{\widetilde{P}(\alpha)}, \dots, \widetilde{K}_{\widetilde{P}(\alpha)+6}}, \widetilde{K}_{\widetilde{P}(\alpha)+7}$$

$$v_\beta = \overbrace{\widetilde{K}_{\widetilde{P}(\beta)}, \dots, \widetilde{K}_{\widetilde{P}(\beta)+6}}, \widetilde{K}_{\widetilde{P}(\beta)+7}$$

$$v_\gamma = \overbrace{\widetilde{K}_{\widetilde{P}(\gamma)}, \dots, \widetilde{K}_{\widetilde{P}(\gamma)+6}}, \widetilde{K}_{\widetilde{P}(\gamma)+7}$$

Outline

Step 1: Reduction of the Number of Systems

$$\mathcal{E}_1 = \{\alpha \text{ odd } | v_{\alpha\{0\to 6\}} = v_0 \gg 1\}$$

$$\mathcal{E}_2(x) = \{\beta \; | v_{\beta\{0\to 6\}} = (v_0 \gg 2, x)\}$$

$$\mathcal{E}_3(x,y) = \{\gamma \text{ odd } | v_{\gamma\{0\to 6\}} = (v_0 \gg 3, x, y)\}$$

Remark:

- Both α and γ are odd.

$$|\mathcal{E}_1| \sim \frac{1}{2}\frac{2^{15}}{2^7} = 2^7 \quad |\mathcal{E}_2(x)| \sim \frac{2^{15}}{2^7} = 2^8 \quad |\mathcal{E}_3(x,y)| \sim \frac{1}{2}\frac{2^{15}}{2^7} = 2^7$$

Thus we have $\sim 2^{22}$ triples (α, β, γ).

Building cost: 7.2^{κ} masks and comparisons.

Step 2: Filters the (a, b, c)

For all $\alpha \in \mathcal{E}_1$
  $x_\alpha \leftarrow$ lsb bit of v_α
  For all $\beta \in \mathcal{E}_2(x_\alpha)$
    $y_\beta \leftarrow$ lsb bit of v_β
    For all $\gamma \in \mathcal{E}_3(x_\alpha, y_\beta)$
      If the system (S) has a solution (a, b, c) in $\mathbb{Z}/2^\kappa\mathbb{Z}$
        $\mathcal{L} \leftarrow \mathcal{L} \cup \{(a, b, c)\}$.

$$S : \begin{pmatrix} \alpha^3 & \alpha^2 & \alpha \\ \beta^3 & \beta^2 & \beta \\ \gamma^3 & \gamma^2 & \gamma \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} + \begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{pmatrix} 2^\kappa$$

Cost of building $\mathcal{L}$: 2^{22} solving of 3×3 systems (~ 25 multiplications each).

Size of $\mathcal{L}$: $\sim 2^{21}$.

Step 3: Rebuilding the key K

- For every solution, we build the corresponding key.
- If this works, we then search for d.

Cost: 2^{κ} AES.

Complexity

Thus the cost of the attack is:

$$2^{15} \text{ AES} + \sim 2^{25} \text{ multiplications on } 16 \text{ bits}.$$

With only half a page of plain text/cipher text, we are able to recover the key K and the whole set of polynomials in less than 10 minutes on a standard PC.

Outline

Simulation Results

Simulations of 300 attacks on 2 distinct computers (150 on each), only using a single core of each computer.

Table: Time of the attacks (in seconds)

Type of processor	Time (FCE)		
	Min.	Max.	Av.
Intel(R) Core(TM)2 Duo CPU E6850 @ 3.00GHz	283 s	784 s	514 s
Intel(R) Xeon(R) CPU 5120 @ 1.86GHz	479 s	1295 s	828 s

FCE Variant

In FCE, computations are made in the ring $\mathbb{Z}/2^{\kappa}\mathbb{Z}$.

For a FCE variant, where the computations are made in the field $GF\left(2^{\kappa}\right)$, the attack can be adapted:

- no trick to get rid of d: naïve attack complexity multiplied by 2^{16},
- complexity of our attack multiplied by 2^{12}.

Conclusion

From a cryptographic point of view, database encryption is still an open problem.