TELECOM ITALIA

Technology – Information Technology – Technical Security

August 18, 2009

Format- and Syntax-Preserving ECB Encryption: Dream or Reality?

| JOVAN GOLIĆ | Security Innovation |

Rump Session of Crypto 2009, Santa Barbara, CA, August 16-20, 2009

TELECOM ITALIA

Outline

1. Objectives
2. Applications
3. Previous Work and Problems
4. Solution for Format-Preserving ECB Encryption
5. Solution for Syntax-Preserving ECB Encryption

A. Format-Preserving Encryption/Decryption Based on RC4-*N*
B. Syntax-Preserving Encryption Based on RC4-*N*
C. Syntax-Preserving Decryption Based on RC4-*N*
D. Initialization Algorithm for RC4-*N*

1. Objectives

- *Symmetric-key encryption:* to be used in the electronic codebook (ECB) mode of operation
- *Length-preserving encryption:* preserves variable plaintext lengths
- *Format-preserving encryption:* preserves length and variable alphabet sizes of plaintext symbols (stateless syntax rules)
- *Syntax-preserving encryption:* preserves syntax rules satisfied by plaintext (stateless or stateful, algorithmically decidable)
- *Secure ECB encryption:* encryption/decryption based on any number of known plaintext/ciphertext pairs is infeasible
- ***Dream or reality?***

2. Applications

- *Selective encryption of sensitive fields in database records*
- *Selective encryption of sensitive data in network traffic records, for traffic analysis (anonymization)*
- *Content encryption of encoded multimedia files for DRM, e.g., in JPEG 2000 and MPEG-4 standards*
- Format and/or syntax compliance disrupts neither testing nor software applications on selectively encrypted data
- ECB encryption induces one-to-one correspondence between original and encrypted data, which enables statistical analysis on selectively encrypted data

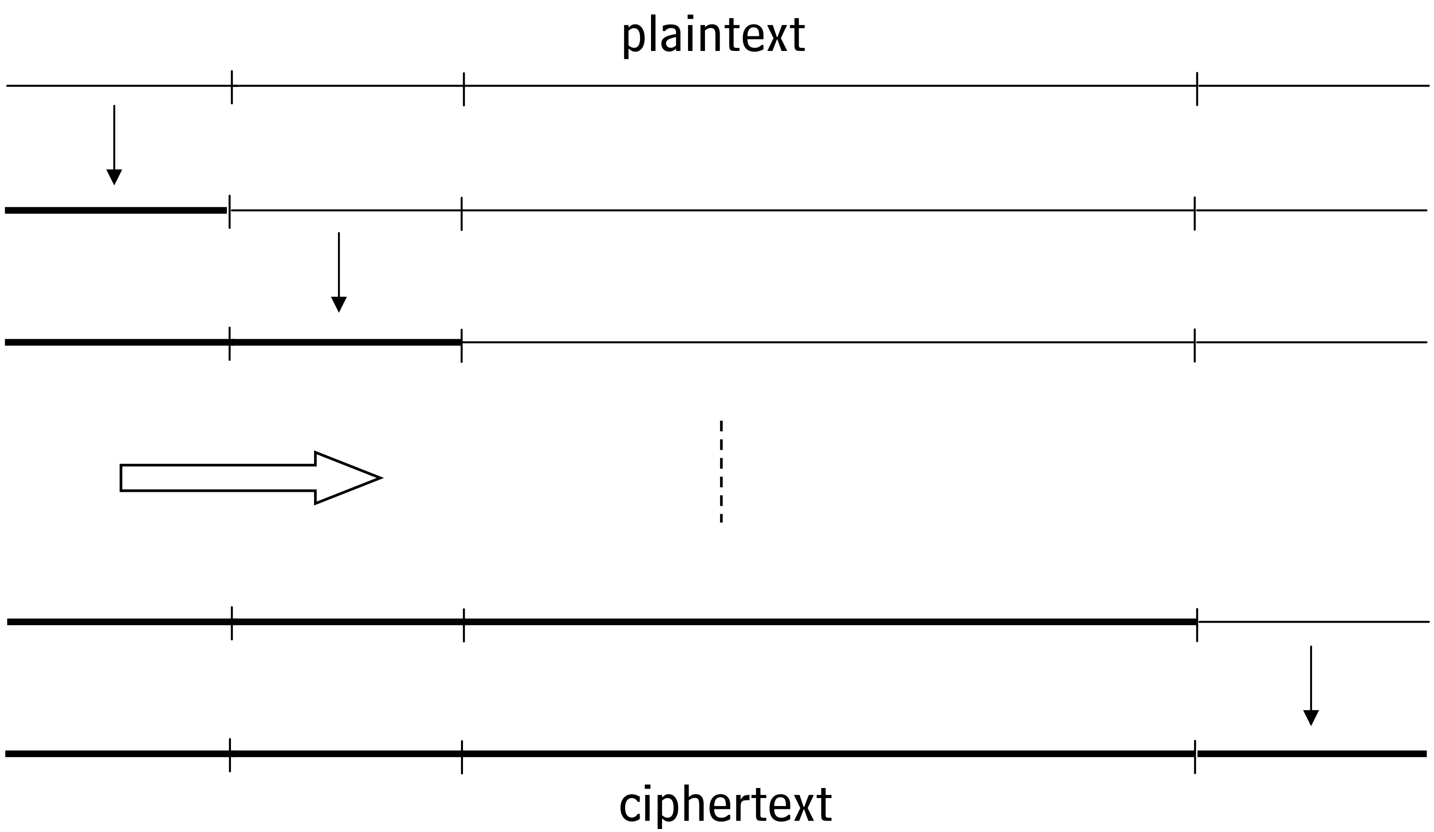
3. Previous Work and Problems-1

- **Format preservation:** map symbols into integers and use modular arithmetic with adapted modulus
- **Patent application US-A-2008-0170693:** a 3-round generic Feistel cipher for ECB encryption, with modular addition and adapted modulus, instead of bitwise XOR:
  - Not flexible w.r.t. length preservation
  - **Adapted modulus disrupts uniform distribution of pseudorandom function output, which results in a statistical distinguisher**
- **Length preservation and ECB encryption:** use symbol-based encryption (stream cipher, but not as keystream generator) **Golić SAC '00**

3. Previous Work and Problems-2

- **Syntax preservation and ECB encryption:** encrypt parts of plaintext one at a time, each time repeating the encryption a *minimum number* of times so that the intermediate ciphertext, composed of all encrypted or unencrypted parts, satisfies the syntax rules; *in decryption, ciphertext parts are decrypted in the reverse order* **Patent application US 2006/0227965 A1:**
  - **Information leakage, because partial plaintexts in combination with partial ciphertexts satisfy syntax rules**
  - Symbol-based parts minimize computation overhead and enable length preservation; proposed encryption is modular addition with a keystream symbol from a conventional stream cipher
  - **ECB encryption is then insecure and range of encrypted symbols need not be maximal possible**

3. Previous Work and Problems-3



4. Solution for Format-Preserving ECB Encryption

- **Use any secure stream cipher** with the range size N of keystream symbols satisfying $N \geq \max_i N_i$, where N_i is the alphabet size of the i-th plaintext symbol (e.g., RC4-N)
- **Combine keystream and plaintext symbols by modular addition,** with the plaintext symbol alphabet size N_i as modulus
- Use **stream cipher with plaintext memory (SCPM) mode,** where each **current state depends on the preceding plaintext symbol**
- **Irregular clocking:** for each plaintext symbol, **additionally update the state a minimum number of times so that the current keystream symbol is uniformly distributed w.r.t. N_i**
- **Apply SCPM mode for (at least) 3 rounds, each time reversing the order of intermediate ciphertext symbols**

5. Solution for Syntax-Preserving ECB Encryption

- Use **stream cipher with ciphertext memory (SCCM) mode,** where each **current state depends on the preceding ciphertext symbol,** *since decryption needs to be run backwards*
- **Irregular clocking:** for each ciphertext symbol, **additionally update the state a minimum number of times so that the current keystream symbol is uniformly distributed w.r.t. N_l as well as either equal to 0 or coprime to N_l**
- **Repeatedly encrypt each symbol to satisfy the syntax rules**
- **Apply SCCM mode for (at least) 3 rounds, each time reversing the order of intermediate ciphertext symbols**
- **Average computation time** roughly proportional to $\sum_l 1/p_l$, where p_l is the average probability that the l-th random symbol is syntax-compliant with the other given plaintext symbols

A. Format-Preserving Encryption/Decryption Based on RC4-N

- Plaintext sequence X, ciphertext sequence Y, keystream sequence Z
- **Initialization:** $S=S(K)$ and $i, j=0$; initial plaintext symbol $x_0=0$
- **Loop for encrypting a plaintext symbol x_l, $l \geq 1$ (one round):**

$m \leftarrow x_{l-1}$

Repeat until $z < N - N \bmod N_l$

$i \leftarrow i + 1$

$j \leftarrow j + S[i] + m$

Swap $S[i], S[j]$

$z \leftarrow S[S[i] + S[j]]$

$m \leftarrow 0$

Output $y_l \leftarrow (x_l + z) \bmod N_l$ *(encryption)*

$x_l \leftarrow (y_l - z) \bmod N_l$ *(decryption)*

B. Syntax-Preserving Encryption Based on RC4-N

- **Initialization:** $S=S(K)$ and $i, j=0$; initial ciphertext symbol $y_0=0$
- Loop for encrypting a plaintext symbol x_l, $l=1,2,\dots,L$ (one round):

$m \leftarrow y_{l-1}$

Repeat until $z < N - N \bmod N_l$ and ($z = 0$ or $\gcd(z, N_l) = 1$)

$i \leftarrow i + 1$

$j \leftarrow j + S[i] + m$

Swap $S[i], S[j]$

$z \leftarrow S[S[i] + S[j]]$

$m \leftarrow 0$

$y \leftarrow x_l$

Repeat until $y_1 y_2 \dots y_{l-1}\, y\, x_{l+1} \dots x_L$ is syntax compliant*

$y \leftarrow (y + z) \bmod N_l$

Output $y_l \leftarrow y$

C. Syntax-Preserving Decryption Based on RC4-*N*

- Loop for generating a keystream symbol z_l, $l=1,2,\ldots,L$ (one round):

$m \leftarrow y_{l-1}$

Repeat until $z < N - N \bmod N_l$ and ($z = 0$ or $\gcd(z, N_l) = 1$)

$i \leftarrow i + 1$

$j \leftarrow j + S[i] + m$

Swap $S[i], S[j]$

$z \leftarrow S[S[i] + S[j]]$

$m \leftarrow 0$

Output and store $z_l \leftarrow z$

- Loop for decrypting a ciphertext symbol y_l, $l=L,L-1,\ldots,1$ (one round):

$x \leftarrow y_l$

Repeat until $y_1 y_2 \ldots y_{l-1}\, x\, x_{l+1} \ldots x_L$ is syntax compliant*

$x \leftarrow (x - z_l) \bmod N_l$

Output $x_l \leftarrow x$

D. Initialization Algorithm for RC4-*N*

- Define $K=(k_0 \ldots k_{N-1})$ from secret key (and round number)
- Set S as $S[i]=i$, $0 \leq i \leq N-1$, and $j=0$
- For $i = 0, \ldots, N-1$

  $j \leftarrow j + S[i] + k_i$

  Swap $S[i], S[j]$
- For $i = 0, \ldots, N-1$

  $j \leftarrow j + S[i]$

  Swap $S[i], S[j]$

  Output $z_i \leftarrow S[S[i] + S[j]]$
- Reset S and j
- For $i = 0, \ldots, N-1$

  $j \leftarrow j + S[i] + z_i$

  Swap $S[i], S[j]$
- Output $S(K) \leftarrow S$