The low-call diet: Authenticated Encryption for call counting HSM users

Mike Bond[1] George French[2] Nigel P. Smart[3] *Gaven J. Watson*[3]

[1]Cryptomathic [2]Barclays Bank Plc. [3]University of Bristol

CT-RSA – March 1st 2013

Setting

- Industry commonly manages keys with special purpose hardware:

Setting

- Industry commonly manages keys with special purpose hardware:
    - Hardware Security Module (HSM).

Setting

- Industry commonly manages keys with special purpose hardware:
  - Hardware Security Module (HSM).
- HSMs store keys which should not be exposed outside the module.

Setting

- Industry commonly manages keys with special purpose hardware:
  - Hardware Security Module (HSM).
- HSMs store keys which should not be exposed outside the module.
- Keys used via an API call to the HSM.

Setting

- Industry commonly manages keys with special purpose hardware:
  - Hardware Security Module (HSM).
- HSMs store keys which should not be exposed outside the module.
- Keys used via an API call to the HSM.
  - e.g. Provides an API call for CBC Mode.
  - Input: plaintext and the name of a key.
  - HSM recovers key and applies CBC-Mode.

Setting

- Industry commonly manages keys with special purpose hardware:
  - Hardware Security Module (HSM).
- HSMs store keys which should not be exposed outside the module.
- Keys used via an API call to the HSM.
  - e.g. Provides an API call for CBC Mode.
  - Input: plaintext and the name of a key.
  - HSM recovers key and applies CBC-Mode.
- Whole process is expensive.

Setting

- Industry commonly manages keys with special purpose hardware:
  - Hardware Security Module (HSM).
- HSMs store keys which should not be exposed outside the module.
- Keys used via an API call to the HSM.
  - e.g. Provides an API call for CBC Mode.
  - Input: plaintext and the name of a key.
  - HSM recovers key and applies CBC-Mode.
- Whole process is expensive.
- *Minimizing* calls to the HSM is important.

What are our options?

Constructions which provide authenticated encryption:

What are our options?

Constructions which provide authenticated encryption:

- Encrypt-then-MAC

What are our options?

Constructions which provide authenticated encryption:

- Encrypt-then-MAC
- Dedicated AE scheme: OCB, EAX, CCM etc.

What are our options?

Constructions which provide authenticated encryption:

- Encrypt-then-MAC
- Dedicated AE scheme: OCB, EAX, CCM etc.

Why not use one of these well studied schemes?

- HSMs designed *before* need for AE was understood.

- HSMs designed *before* need for AE was understood.
- More modern modes are not supported.

- HSMs designed *before* need for AE was understood.
- More modern modes are not supported.

Solution:

Use a generic construction such as Encrypt-then-MAC.

- HSMs designed *before* need for AE was understood.
- More modern modes are not supported.

Solution:

Use a generic construction such as Encrypt-then-MAC.

Solution Problem:

- This uses two keys.
- Meaning two HSM calls.

Design criteria

Basic requirements:

Design criteria

Basic requirements:

- All secret keys should reside on the HSM.
- Only one call to the HSM is allowed, i.e. single key.
- Such a call should be to a CBC-Encrypt.

Encryption with redundancy

- Studied formally by An and Bellare.

Encryption with redundancy

- Studied formally by An and Bellare.
- Two types of redundancy function; secret key and public key.

Encryption with redundancy

- Studied formally by An and Bellare.
- Two types of redundancy function; secret key and public key.
- IND-CPA encryption scheme + secret/public redundancy function $\not\Rightarrow$ AE.

Encryption with redundancy

- Studied formally by An and Bellare.
- Two types of redundancy function; secret key and public key.
- IND-CPA encryption scheme + secret/public redundancy function $\not\Rightarrow$ AE.
- An and Bellare define a scheme with a secret key redundancy function, Nested CBC (NCBC).
- NCBC uses a *different* key to encrypt the last block.

Relating to our scheme

- Our scheme uses secret redundancy,

Relating to our scheme

- Our scheme uses secret redundancy,
  where the redundancy function uses a different “key” each time.

Relating to our scheme

- Our scheme uses secret redundancy,
  where the redundancy function uses a different “key” each time.
- In general any IND-CPA scheme plus one time redundancy function $\not\Rightarrow$ AE.

API call

- The API call is CBC-mode

API call

- The API call is CBC-mode with all-zero IV.

API call

- The API call is CBC-mode with all-zero IV.
- Need randomness for security.

API call

- The API call is CBC-mode with all-zero IV.
- Need randomness for security.
- Use HSMs ability to generate random numbers.

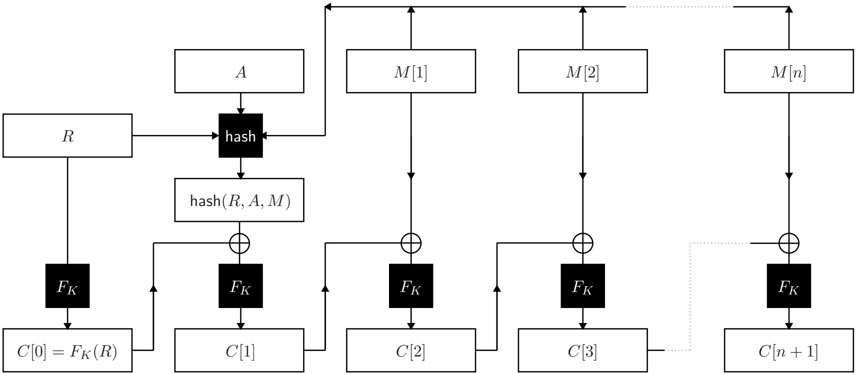
API call

- The API call is CBC-mode with all-zero IV.
- Need randomness for security.
- Use HSMs ability to generate random numbers.
- *Implementation note* – to avoid making an extra HSM call for every encryption, we maintain a cache of randomness.

API call

- The API call is CBC-mode with all-zero IV.
- Need randomness for security.
- Use HSMs ability to generate random numbers.
- *Implementation note* – to avoid making an extra HSM call for every encryption, we maintain a cache of randomness.
- We assume this cache to be secure.

Managed Encryption Format



$\underline{\text{Encrypt}(K, A, M)}$	$\underline{\text{Decrypt}(K, A, C)}$
$R \xleftarrow{r} \{0,1\}^l$ $H \leftarrow \text{hash}(R, A, M)$ $C \leftarrow \text{E-CBC}[F](K, R\|H\|\text{pad}(M))$ **return** C	$R\|H\|M' \leftarrow \text{D-CBC}[F](K, C)$ $M \leftarrow \text{dpad}(M')$ **if** $M \neq \perp$ **then** $\quad \overline{h} \leftarrow \text{hash}(R, A, M)$ **if** $\overline{h} \neq h$ **then** $M = \perp$ **return** M

Encrypt(K, A, M)	Decrypt(K, A, C)
$R \xleftarrow{r} \{0,1\}^l$ $H \leftarrow \text{hash}(R, A, M)$ $C \leftarrow \text{E-CBC}[F](K, R \| H \| \text{pad}(M))$ **return** C	$R \| H \| M' \leftarrow \text{D-CBC}[F](K, C)$ $M \leftarrow \text{dpad}(M')$ **if** $M \neq \perp$ **then** $\overline{h} \leftarrow \text{hash}(R, A, M)$ **if** $\overline{h} \neq h$ **then** $M = \perp$ **return** M

Points to note:

- Padding (uniform error reporting)
- "MAC-then-encrypt"
- IV

Security model – Privacy

Let $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a symmetric encryption scheme.

$$\begin{array}{|l|}\hline \underline{\mathbf{Enc}(A, M_0, M_1)} \\ C_0 \leftarrow \mathsf{Encrypt}(K, A, M_0) \\ C_1 \leftarrow \mathsf{Encrypt}(K, A, M_1) \\ \mathcal{C} \stackrel{\cup}{\leftarrow} C_b \\ \mathbf{return}\ C_b \\ \hline \end{array}$$

$$\begin{array}{l} \underline{\mathbf{PRIV}^{\mathcal{A}}(\Pi)} \\ K \leftarrow \mathsf{KeyGen};\ b \stackrel{r}{\leftarrow} \{0,1\} \\ b' \leftarrow \mathcal{A}^{\mathbf{Enc}} \\ \mathbf{return}\ (b' = b) \end{array}$$

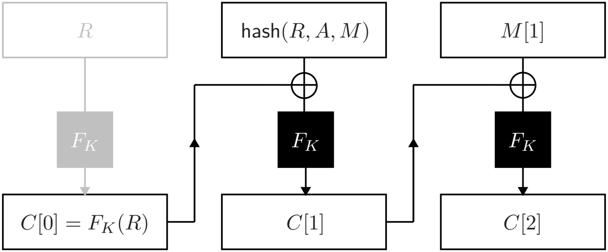
$$\mathbf{Adv}_{\Pi}^{\mathrm{priv}}(\mathcal{A}) = 2\Pr[\mathbf{PRIV}^{\mathcal{A}}(\Pi) \Rightarrow \mathsf{true}] - 1,$$

PRIV

This can be proved by relating to the security of CBC mode proved by Bellare et al. [BDJR].

PRIV

This can be proved by relating to the security of CBC mode proved by Bellare et al. [BDJR].



Privacy

- Let $F = \{F_K : K \in \{0,1\}^k\}$ be a permutation family.
- Let $\Pi[F]$ be the managed encryption format using permutation family F.
- Let $\mathcal{A}$ be an adversary against Privacy which runs in time t; making q_e encryption queries totalling at most μ_e bits.

Privacy

- Let $F = \{F_K : K \in \{0,1\}^k\}$ be a permutation family.
- Let $\Pi[F]$ be the managed encryption format using permutation family F.
- Let $\mathcal{A}$ be an adversary against Privacy which runs in time t; making q_e encryption queries totalling at most μ_e bits.

Then there exists adversary $\mathcal{B}$ such that:

$$\mathbf{Adv}^{\mathrm{PRIV}}_{\Pi[F]}(\mathcal{A}) \leq 2\mathbf{Adv}^{\mathrm{prp}}_{F}(\mathcal{B}) + \frac{q_f^2}{2^l} + \frac{1}{2^l}\left(\left(\frac{\mu_e}{l} + 2q_e\right)^2 - \left(\frac{\mu_e}{l} + 2q_e\right)\right)$$

where $\mathcal{B}$ runs in time $t + O(\mu_e)$ asking at most $q_f = \frac{\mu_e}{l} + 2q_e$ queries.

Security model – AUTH

Let $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a symmetric encryption scheme.

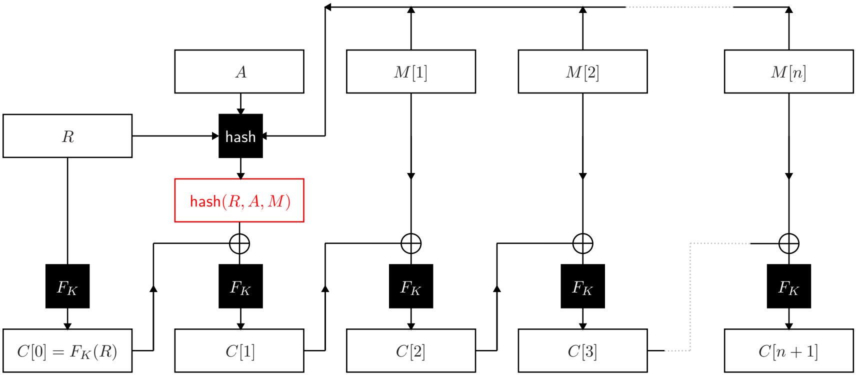
Enc(A, M)	**Test**(A^*, C^*)
$C \leftarrow \mathsf{Encrypt}(K, A, M)$ $\mathcal{C} \xleftarrow{\cup} (A, C)$ **return** C	$M^* \leftarrow \mathsf{Decrypt}(K, A^*, C^*)$ **if** $M^* \neq \perp$ and $(A^*, C^*) \notin \mathcal{C}$ **then** win $\leftarrow$ true **return** $(M^* \neq \perp)$

AUTH$^{\mathcal{A}}(\Pi)$

$K \leftarrow \mathsf{KeyGen}$
win $\leftarrow$ false
$(A^*, C^*) \leftarrow \mathcal{A}^{\textbf{Enc}, \textbf{Test}}$
return win

$$\mathbf{Adv}_{\Pi}^{\mathrm{auth}}(\mathcal{A}) = \Pr[\textbf{AUTH}^{\mathcal{A}}(\Pi) \Rightarrow \mathsf{true}]$$

AUTH



To forge a ciphertext the adversary must forge the hash.

Case 1: Hash not queried

$$\Pr[(\text{hash}(R^*, A^*, M^*) = h^*) \wedge ((R^*, A^*, M^*, h^*) \notin \mathcal{H}) | \pi \xleftarrow{r} \text{Perm}] \leq \frac{q_t}{2^l}$$

- Not previously queried.
- Random chance on verification.

Case 2: Hash already queried

$$\Pr[(\text{hash}(R^*, A^*, M^*) = h^*) \wedge ((R^*, A^*, M^*, h^*) \in \mathcal{H}) | \pi \xleftarrow{r} \mathrm{Perm}] \leq \frac{q_h \mu_e}{l 2^l}.$$

- Previous call to random oracle.
- If call made by encryption query then invalid forgery.
- So independent call to hash.

Case 2: Hash already queried

$$\Pr[(\mathsf{hash}(R^*, A^*, M^*) = h^*) \wedge ((R^*, A^*, M^*, h^*) \in \mathcal{H}) | \pi \xleftarrow{r} \mathrm{Perm}] \leq \frac{q_h \mu_e}{l 2^l}.$$

- Previous call to random oracle.
- If call made by encryption query then invalid forgery.
- So independent call to hash.
- Analysis is then based on the collision event that for some i, j,

$$C_i[j] \oplus M_i[j] = h^* \oplus \pi(R^*).$$

AUTH

- Let $F = \{F_K : K \in \{0,1\}^k\}$ be a permutation family.
- Let $\Pi[F]$ be the managed encryption format using permutation family F.
- Let $\mathcal{A}$ be an adversary against the AUTH security which runs in time t; making q_e encryption queries totalling at most μ_e bits, q_t test queries totalling at must μ_t bits and q_h random oracle queries.

AUTH

- Let $F = \{F_K : K \in \{0,1\}^k\}$ be a permutation family.
- Let $\Pi[F]$ be the managed encryption format using permutation family F.
- Let $\mathcal{A}$ be an adversary against the AUTH security which runs in time t; making q_e encryption queries totalling at most μ_e bits, q_t test queries totalling at must μ_t bits and q_h random oracle queries.

Then there exists adversary $\mathcal{B}$ such that:

$$\mathbf{Adv}^{\mathrm{AUTH}}_{\Pi[F]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{sprp}}_{F}(\mathcal{B}) + \frac{q_t}{2^l} + \frac{q_h \mu_e}{l 2^l}$$

where $\mathcal{B}$ makes $q_f = \frac{\mu_e}{l} + 2q_e + \frac{\mu_t}{l}$ queries and runs in time $t + O(\mu_e + \mu_t)$.

Summary

- We have discussed the Managed Encryption Format

Summary

- We have discussed the Managed Encryption Format
- Despite its limitation we were still able to prove it secure.

Summary

- We have discussed the Managed Encryption Format
- Despite its limitation we were still able to prove it secure.
- With several important implementation caveats.

Summary

- We have discussed the Managed Encryption Format
- Despite its limitation we were still able to prove it secure.
- With several important implementation caveats.
- Care needs to be taken with implementation to ensure security.

Questions

Weak Keys of the Full MISTY1 Block Cipher for Related-Key Differential Cryptanalysis

Jiqiang Lu

Institute for Infocomm Research,
Agency for Science, Technology and Research,
1 Fusionopolis Way, Singapore 138632
jlu@i2r.a-star.edu.sg, lvjiqiang@hotmail.com

Joint work with Wun-She Yap and Yongzhuang Wei.

CT-RSA 2013

Outline:

1. Introduction
2. Related Work
3. A Class of $2^{102.57}$ Weak Keys
4. A 7-Round Related-Key Differential with Prob. 2^{-58}
5. Attacking the Full MISTY1 under the Weak Keys
6. Another Class of $2^{102.57}$ Weak Keys
7. Conclusions

1.1 Block Cipher

- An important primitive in symmetric-key cryptography.
  - * Main purpose: provide confidentiality — A most fundamental security goal.
- An algorithm that transforms a fixed-length data block into another data block of the same length under a secret user key.
  - * Input: plaintext.
  - * Output: ciphertext.
  - * Three sub-algorithms: encryption, decryption, key schedule.
- Constructed by repeating a simple function many times, known as the iterated method.
  - * An iteration: a round.
  - * The repeated function: the round function.
  - * The key used in a round: a round subkey.
  - * The number of iterations: the number of rounds.
  - * The round subkeys are generated from the user key under a key schedule algorithm.

1.2 A Cryptanalytic Attack

- An algorithm that distinguishes a cryptosystem from a random function.
- Usually measured using the following three metrics:
  * Data complexity
    - The numbers of plaintexts and/or ciphertexts required.
  * Memory (storage) complexity
    - The amount of memory required.
  * Time (computational) complexity
    - The amount of computation or time required, how many encryptions/decryptions or memory accesses.
- Goals:
  * Break a cryptosystem (ideally, in a practical complexity).
  * Enable more secure cryptosystems to be designed.

1.3 Related-Key (Differential) Cryptanalysis

- Independently introduced by Knudsen in 1992 and Biham in 1993.
- Different from differential cryptanalysis: The pair of ciphertexts are obtained by encrypting the pair of plaintexts using two different keys with a particular relationship, e.g. certain difference.
- Probability of a related-key differential:

$$\Pr_{\mathbb{E}_K,\mathbb{E}_{K'}}(\Delta\alpha \to \Delta\beta) = \Pr_{P\in\{0,1\}^n}(\mathbb{E}_K(P) \oplus \mathbb{E}_{K'}(P \oplus \alpha) = \beta).$$

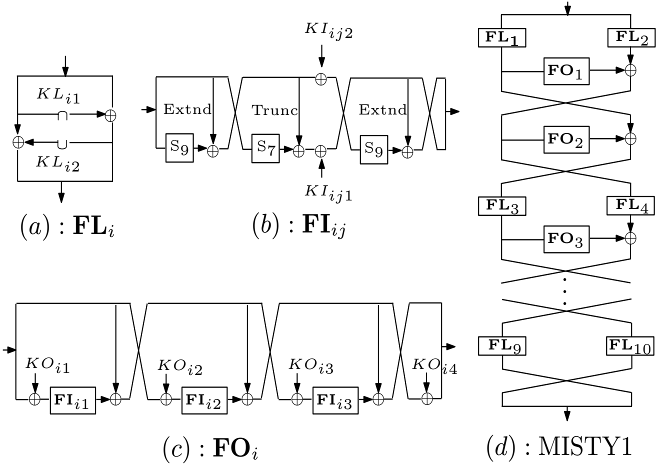
- For a random function, the expected probability of any related-key differential is 2^{-n}.

If $\Pr_{\mathbb{E}_K,\mathbb{E}_{K'}}(\Delta\alpha \to \Delta\beta) > 2^{-n}$, we can use the related-key differential to distinguish $\mathbb{E}$ from a random function.

1.4.1 Introduction

- Designed by Mitsubishi (Matsui et al.), published in 1995.
- A 64-bit block cipher, a user key of 128 bits, and a recommended number of 8 rounds, with a total of 10 key-dependent logical functions **FL**:
  - two **FL** functions at the beginning;
  - two **FL** functions inserted after every two rounds.
- A Japanese CRYPTREC-recommended e-government cipher, an European NESSIE selected cipher, an ISO international standard.
- Widely used in Mitsubishi products as well as in Japanese military.

1.4.2 Structure



1.4.3 Key Schedule

1. Represent a user key K as eight 16-bit words $K = (K_1, K_2, \cdots, K_8)$.
2. Generate a different set of eight 16-bit words $K'_1, K'_2, \cdots, K'_8$ by

$$K'_i = \mathbf{FI}(K_i, K_{i+1}), \text{ for } i = 1, 2, \cdots, 8.$$

3. Subkeys:

$$KO_{i1} = K_i, KO_{i2} = K_{i+2}, KO_{i3} = K_{i+7}, KO_{i4} = K_{i+4};$$
$$KI_{i1} = K'_{i+5}, KI_{i2} = K'_{i+1}, KI_{i3} = K'_{i+3};$$
$$KL_i = K_{\frac{i+1}{2}} || K'_{\frac{i+1}{2}+6}, \text{ for } i = 1, 3, 5, 7, 9; \text{otherwise}, KL_i = K'_{\frac{i}{2}+2} || K_{\frac{i}{2}+4}.$$

1.4.4 Security

- Has been extensively analysed against a variety of cryptanalytic methods.
- No whatever cryptanalytic attack on the full version.

2. Related Work

Dai and Chen's related-key differential attack on 8-round MISTY1 with only the last 8 **FL** functions (INSCRYPT 2011).

- A class of 2^{105} weak keys.
  - A weak key is a user key under which a cipher is more vulnerable to be attacked.
- A 7-round related-key differential characteristic with probability 2^{-60}.
- Attacking the 8-round reduced version under weak keys.
  - Attack procedure is straightforward, by conducting a key recovery on $\mathbf{FO}_1$ in a way similar to the early abort technique for impossible differential cryptanalysis.
  - Data complexity: 2^{63} chosen ciphertexts.
  - Memory complexity: 2^{35} bytes.
  - Time complexity: $2^{86.6}$ encryptions.

2.1 A Class of 2^{105} Weak Keys

Three binary constants:

* 7-bit $a = 0010000$;
* 16-bit $b = 0010000000010000$;
* 16-bit $c = 0010000000000000$.

Let K_A, K_B be two 128-bit user keys:

$$K_A = (K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8),$$
$$K_B = (K_1, K_2, K_3, K_4, K_5, K_6^*, K_7, K_8).$$

Let K'_A, K'_B be the corresponding 128-bit words generated by the key schedule:

$$K'_A = (K'_1, K'_2, K'_3, K'_4, K'_5, K'_6, K'_7, K'_8),$$
$$K'_B = (K'_1, K'_2, K'_3, K'_4, K'^*_5, K'^*_6, K'_7, K'_8).$$

The class of weak keys is defined to be the set of all possible (K_A, K_B) satisfying the following 10 conditions:

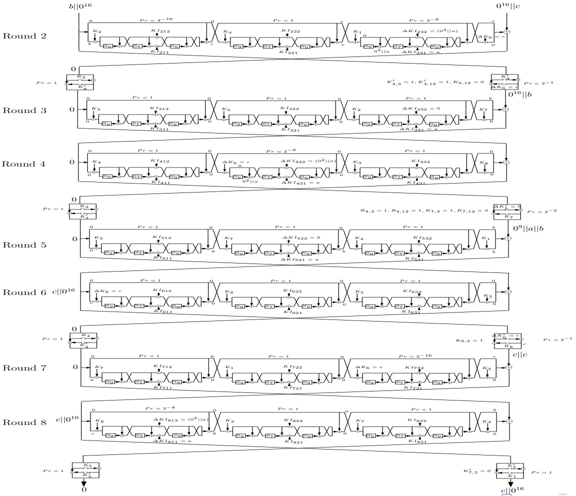
$$K_6 \oplus K_6^* = c, \quad K'_5 \oplus K'^*_5 = b, \quad K'_6 \oplus K'^*_6 = c, \quad K_{6,12} = 0, \quad K_{7,3} = 1,$$
$$K_{7,12} = 0, \quad K_{8,3} = 1, \quad K'_{4,3} = 1, \quad K'_{4,12} = 1, \quad K'_{7,3} = 0.$$

The number:

$$|K_1| = 2^{16}, |K_2| = 2^{16}, |K_3| = 2^{16}, |(K_4, K_5)| = 2^{30}, |(K_6, K_7, K_8)| = 2^{27}.$$

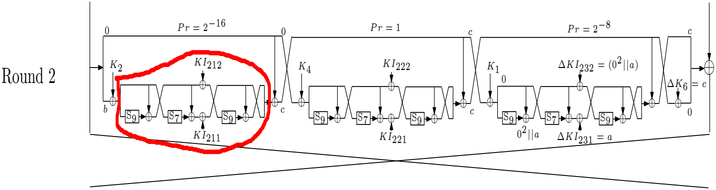
Therefore, a total of 2^{105} weak keys.

2.2 A 7-Round Related-Key Differential Characteristic



3. A Class of $2^{102.57}$ Weak Keys

Focus on the 7-round related-key differential characteristic.

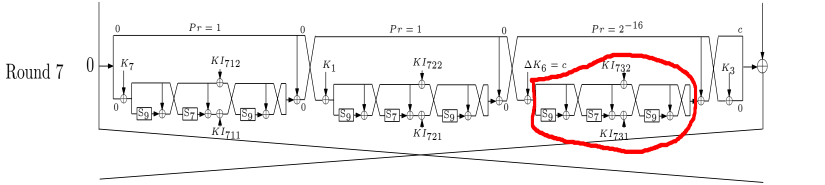


Not all the 2^{15} possible K_7' (i.e. KI_{21}) defined by the weak key class make $\Pr_{\mathbf{FI}_{21}}(\Delta b \to \Delta c) > 0$!

The number of K_7' defined by the weak key class is 2^{15}, the number of K_7' satisfying $\Pr_{\mathbf{FI}_{21}}(\Delta b \to \Delta c) > 0$ is about $2^{14.57}$.

The number of K_7' defined by the weak key class & satisfying $\Pr_{\mathbf{FI}_{21}}(\Delta b \to \Delta c) > 0$ is about $2^{13.57}$.

$\Pr_{\mathbf{FI}_{21}}(\Delta b \to \Delta c) = 2^{-15}/2^{-14}/2^{-13.42}$.



Not all the 2^{16} possible K'_2 (i.e. KI_{73}) defined by the weak key class make $\Pr_{\mathbf{FI}_{73}}(\Delta c \to \Delta c) > 0$!

The number of K'_2 defined by the weak key class is 2^{16}, the number of K'_2 satisfying $\Pr_{\mathbf{FI}_{21}}(\Delta b \to \Delta c) > 0$ is 2^{15}.

The number of K'_2 defined by the weak key class & satisfying $\Pr_{\mathbf{FI}_{73}}(\Delta c \to \Delta c) > 0$ is 2^{15}.

$\Pr_{\mathbf{FI}_{73}}(\Delta c \to \Delta c) = 2^{-15}$.

As a result, a class of $2^{102.57}$ weak keys:

$$|K_1| = 2^{16}, |(K_2, K_3)| = 2^{31}, |(K_4, K_5)| = 2^{30}, |(K_6, K_7, K_8)| \approx 2^{25.57}.$$

* $|K_3| = 2^{16}$, $|K_5| = 2^{16}$.
* $|K'_7| = 2^{13.57}$; $\forall K'_7$, $\exists\ 2^{12}$ (K'_6, K_8).
* $|K'_{2,8-16}| = 2^8$, $|K'_3| = 2^{16}$, $|K'_{4,8-16}| = 2^8$.

4. A 7-Round Related-Key Differential with Prob. 2^{-58}

A 7-round related-key differential with probability 2^{-58}.

$$(b||0^{32}||c) \rightarrow (0^{32}||c||0^{16}).$$

5.1 Precomputation

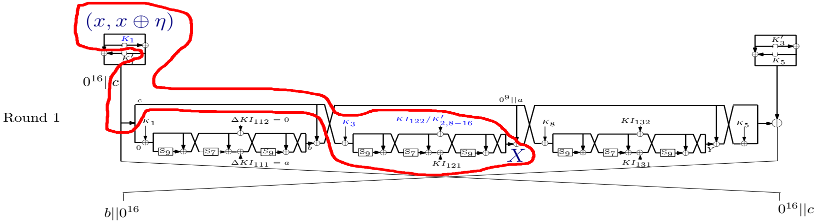
Hash table $\mathcal{T}_1$:

$(x, x \oplus \eta)$: The left halves of a plaintext pair

Only three possible input differences $\eta = \overbrace{00?0000000000000||00?0000000000000}^{32\ bits}$

X: output difference of $\mathbf{FI}_{12}$

Store satisfying $(K_1, K_3, K'_{2,8-16})$ into Table $\mathcal{T}_1$ indexed by (x, η, X)



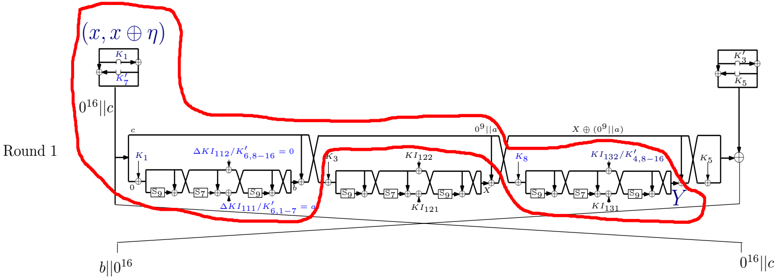
Memory complexity: $2^{75.91}$ bytes; Time complexity: $2^{73.59}$ **FI** computations.

For every (x, η, X), there are 2^{23} satisfying $(K_1, K_3, K'_{2,8-16})$ on average.

Hash table $\mathcal{T}_2$:

Y: output difference of $\mathbf{FI}_{13}$

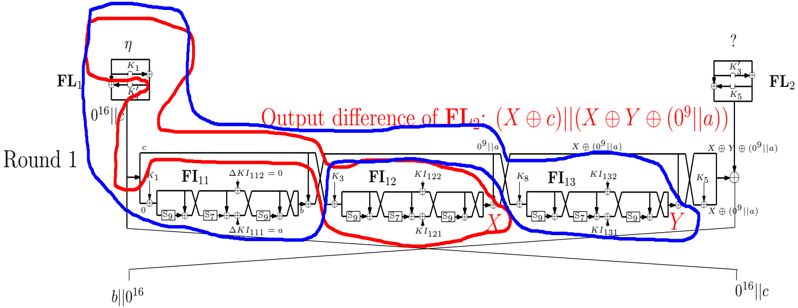
Store satisfying (K_6, K_7, K_8) into Table $\mathcal{T}_2$ indexed by $(x, \eta, Y, K_1, K'_{4,8-16})$



Memory complexity: $2^{84.74}$ bytes; Time complexity: $2^{84.16}$ **FI** computations.

For every $(x, \eta, Y, K_1, K'_{4,8-16})$, there are $2^{9.57}$ satisfying (K_6, K_7, K_8) on average.

5.2 Attack Outline



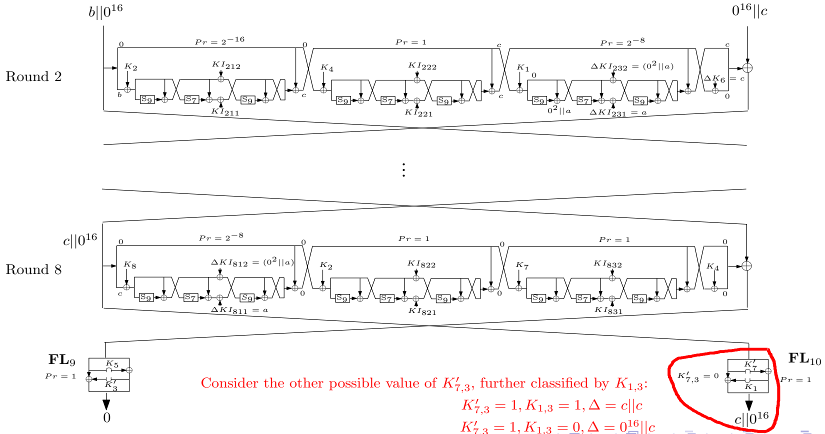
Step 1: Choose 2^{60} ciphertext pairs with difference $(0^{32}||c||0^{16})$.
Step 2: Keep plaintext pairs with difference $(\eta||?)$
Step 3: Focus on $\mathbf{FL}_2$. Guess (K_3', K_5), compute X, Y.
Step 4: Focus on $\mathbf{FL}_1$ and $\mathbf{FI}_{12}$. Obtain satisfying $(K_1, K_3, K_{2,8-16}')$ from Table $\mathcal{T}_1$.
Step 5: Retrieve K_4 from $K_3' = \mathbf{FI}(K_3, K_4)$, compute $K_4' = \mathbf{FI}(K_4, K_5)$.
Step 6: Focus on $\mathbf{FL}_1$, $\mathbf{FI}_{11}$ and $\mathbf{FI}_{13}$. Obtain satisfying (K_6, K_7, K_8) from Table $\mathcal{T}_2$.
Step 7: Increase 1 to counters for $(K_1, K_{2,8-16}', K_3, K_4, K_5, K_6, K_7, K_8)$.
Step 8: For a subkey guess whose counter number is larger than or equal to 3, exhaustively search the remaining 7 key bits.

5.3 Attack Complexity

- Data complexity: 2^{61} chosen ciphertexts.
- Memory complexity: $2^{99.2}$ bytes.
- Time complexity: $2^{87.94}$ encryptions.
- Success probability: 76%.

6. Another Class of $2^{102.57}$ Weak Keys

Focus on the 7-round related-key differential characteristic:



Consider the other possible value of $K'_{7,3}$, further classified by $K_{1,3}$:

$K'_{7,3} = 1, K_{1,3} = 1, \Delta = c||c$

$K'_{7,3} = 1, K_{1,3} = 0, \Delta = 0^{16}||c$

7. Conclusions

Have presented a related-key differential attack on the full MISTY1 algorithm under certain weak key assumptions.

* Have described $2^{103.57}$ weak keys for a related-key differential attack on the full MISTY1.
* Quite theoretical, for the attack works under the assumptions of weak-key and related-key scenarios and its complexity is very high.

The MISTY1 cipher does not behave like a random function (in the related-key model), and cannot be regarded to be an ideal cipher.

Thank you!

RSA CONFERENCE 2013

A Fully Homomorphic Cryptosystem with Approximate Perfect Secrecy

Michal Hojsík, Veronika Půlpánová

Department of Algebra
Charles University in Prague

Session ID: CRYP-F42

Session Classification: Advanced

Outline

- (Fully) Homomorphic Encryption
- Polly Cracker
- Symmetric Polly Cracker
- Security of SymPC
- Conclusions

Homomorphic Encryption

- Set of plaintexts $\mathcal{P}$, set of ciphertexts $\mathcal{C}$, set of keys $\mathcal{K}$
- For all keys $k \in \mathcal{K}$, encryption e_k, decryption d_k

$$\mathcal{P} \underset{d_k}{\overset{e_k}{\rightleftarrows}} \mathcal{C}$$

- Goal: Calculations on $\mathcal{P} \sim$ calculations on $\mathcal{C}$

Homomorphic Encryption

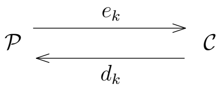
- Set of plaintexts $\mathcal{P}$, set of ciphertexts $\mathcal{C}$, set of keys $\mathcal{K}$
- For all keys $k \in \mathcal{K}$, encryption e_k, decryption d_k

$$\mathcal{P} \underset{d_k}{\overset{e_k}{\rightleftarrows}} \mathcal{C}$$

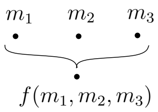
- Goal: Calculations on $\mathcal{P}$ $\sim$ calculations on $\mathcal{C}$

Homomorphic Encryption

- Set of plaintexts $\mathcal{P}$, set of ciphertexts $\mathcal{C}$, set of keys $\mathcal{K}$
- For all keys $k \in \mathcal{K}$, encryption e_k, decryption d_k

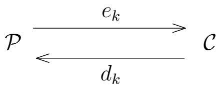


- Goal: Calculations on $\mathcal{P}$ $\sim$ calculations on $\mathcal{C}$

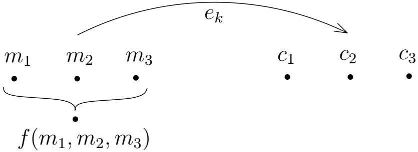


Homomorphic Encryption

- Set of plaintexts $\mathcal{P}$, set of ciphertexts $\mathcal{C}$, set of keys $\mathcal{K}$
- For all keys $k \in \mathcal{K}$, encryption e_k, decryption d_k

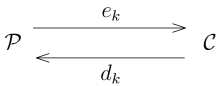


- Goal: Calculations on $\mathcal{P} \sim$ calculations on $\mathcal{C}$

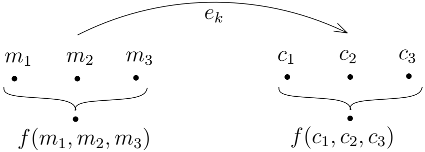


Homomorphic Encryption

- Set of plaintexts $\mathcal{P}$, set of ciphertexts $\mathcal{C}$, set of keys $\mathcal{K}$
- For all keys $k \in \mathcal{K}$, encryption e_k, decryption d_k

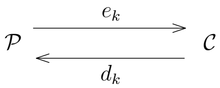


- Goal: Calculations on $\mathcal{P}$ $\sim$ calculations on $\mathcal{C}$

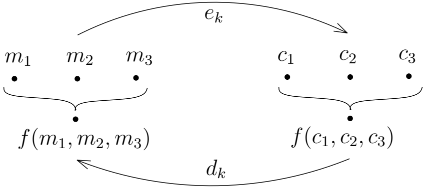


Homomorphic Encryption

- Set of plaintexts $\mathcal{P}$, set of ciphertexts $\mathcal{C}$, set of keys $\mathcal{K}$
- For all keys $k \in \mathcal{K}$, encryption e_k, decryption d_k



- Goal: Calculations on $\mathcal{P} \sim$ calculations on $\mathcal{C}$



Homomorphic Encryption cont.

- Endow $\mathcal{P}, \mathcal{C}$ with operations: $(\mathcal{P}, \cdot), \ (\mathcal{C}, \odot)$
- Cryptosystem is homomorphic if and only if:

  $d_k : (\mathcal{C}, \odot) \rightarrow (\mathcal{P}, \cdot)$ is a homomorphism

  d_k "preserves operation": $d_k(c_1 \odot c_2) = d_k(c_1) \cdot d_k(c_2)$
- e_k may be non-deterministic
- Example - Plain RSA: $(\mathcal{P}, \cdot) = (\mathcal{C}, \cdot) = (\mathbb{Z}_N, \cdot)$

  $$(c_1 \cdot c_2)^d \mod N = (c_1^d \mod N) \cdot (c_2^d \mod N) \mod N$$

  $\rightarrow$ Plain RSA is multiplicatively homomorphic
- Other examples: Goldwasser-Micali, Benaloh: $(\mathcal{P}, +), \ (\mathcal{C}, \cdot)$

Fully Homomorphic Encryption

- One operation $\longrightarrow$ limited applications
- Need more operations on $\mathcal{P}$ and $\mathcal{C}$
- **Fully Homomorphic Cryptosystem**: $(\mathcal{P}, +, \cdot)$, $(\mathcal{C}, \oplus, \odot)$ rings

  $d_k : (\mathcal{C}, \oplus, \odot) \to (\mathcal{P}, +, \cdot)$ is a ring homomorphism
- E.g. for $\mathcal{P} = GF(2^n)$ and $(\mathcal{C}, \oplus, \odot)$ a ring

  $\to$ Homomorphic evaluation of any circuit (Boolean function)

$$f(m_1, \dots, m_r) = d_k\left(f\left(e_k(m_1), \dots, e_k(m_r)\right)\right)$$

Fully Homomorphic Encryption cont.

- Many practical applications
- Outsourcing computations on confidential data
  $\rightarrow$ "encrypted cloud computing"

Various constructions:

- Gentry 2009, lattice-based cryptography with Bootstrapping
- DGHV 2009, modular arithmetic with Bootstrapping
- AAPS 2011, coding theory with limited multiplication
- Fellows, Koblitz 1994, ideal membership problem, Polly Cracker

Polly Cracker

- Probabilistic public-key cryptosystem
- $\mathcal{P} = GF(q) = \mathbb{F}$, $\mathcal{C} = \mathbb{F}[x_1, \ldots, x_n]$
- Private key $\vec{s} \in \mathbb{F}^n$
- Public key $PK = \{f_1, \ldots, f_r\} \subset \mathcal{C}, \ \ \forall i \ \ f_i(\vec{s}) = 0$
- Encryption of $m \in \mathbb{F}$: choose $J \subset \{1, \ldots, r\}$ uniformly at random

$$c = e(m) = m + \sum_{j \in J} f_j$$

- Decryption of $c \in \mathcal{C}$ – evaluation of c at $\vec{s}$:

$$d_{\vec{s}}(c) = c(\vec{s}) = m + \sum_{j \in J} f_j(\vec{s}) = m$$

Polly Cracker cont.

- Fully homomorphic
- Polynomial evaluation is a ring homomorphism
- Let $c_1 = m_1 + \sum_{i \in I} f_i, \quad c_2 = m_2 + \sum_{j \in J} f_j$

$$d(c_1 + c_2) = (c_1 + c_2)(\vec{s}) = \left(m_1 + \sum_{i \in I} f_i + m_2 + \sum_{j \in J} f_j \right)(\vec{s}) = m_1 + m_2$$

$$d(c_1 \cdot c_2) = (c_1 \cdot c_2)(\vec{s}) = \left((m_1 + \sum_{i \in I} f_i)(m_2 + \sum_{j \in J} f_j) \right)(\vec{s}) = m_1 m_2$$

- Attack by calculation of Gröbner basis of the ideal $\langle PK \rangle$ - G
- Decryption of c equals $c \mod \langle G \rangle$

Symmetric Polly Cracker (SymPC)

- Probabilistic symmetric-key cryptosystem
- Secret key $\vec{s} \in \mathbb{F}^n$, $\mathbb{F} = GF(q)$
- Multiplicative key $G = \{g_1, \ldots, g_n\} \subset \mathbb{F}[x_1, \ldots, x_n]$
  used in calculations with ciphertexts (not a public key)
- $\mathcal{P} = \mathbb{F}$, $\mathcal{C} = \mathbb{F}[x_1, \ldots, x_n]/\langle G \rangle$
- G has special properties (G is the reduced Gröbrer basis)
  $\rightarrow$ Easily algorithmized multiplicative structure on $\mathcal{C}$
  $\rightarrow$ Reduces complexity and size of ciphertexts

Symmetric Polly Cracker (SymPC) cont.

- $\mathcal{P} = \mathbb{F},\ \mathcal{C} = \mathbb{F}[x_1, \ldots, x_n]/\langle G \rangle$
- Encryption of $m \in \mathcal{P}$: choose $f \in \mathcal{C}$ uniformly at random

$$e_{\vec{s}}(m) = f - f(\vec{s}) + m$$

- Decryption of $c \in \mathcal{C}$ – evaluation of c at $\vec{s}$:

$$d_{\vec{s}}(c) = c(\vec{s}) = (f - f(\vec{s}) + m)(\vec{s}) = m$$

- Fully homomorphic
- Complexity analysis in the paper

Security of SymPC

Approximate perfect secrecy:

- For all probability distributions on $\mathcal{P}$ and for all $m \in \mathcal{P}$

$$\Pr[P = m \mid C = c] \xrightarrow{t \to \infty} \Pr[P = m]$$

  for almost all $c \in C$ (security parameter t)

- Assuming an attacker with unbounded computational power
- Probabilistic information theoretical security

Security of SymPC cont.

- Approximate perfect secrecy in bounded CPA model
- k-bounded CPA: an attacker can obtain at most k pair (m, c)
- Not CCA secure:

  Ask for decryption of $c_1 = x_1, c_2 = x_2, \ldots, c_n = x_n$

  $\rightarrow$ obtain the secret key $(s_1, s_2, \ldots, s_n) = \vec{s}$ as $c_i(\vec{s}) = x_i(\vec{s})$

- KPA security $\sim$ CPA security:

  For a given $(m, c) \in \mathcal{P} \times \mathcal{C}$ s.t. $c(\vec{s}) = m$ and any $m' \in \mathcal{P}$

  The pair $(m', c' = c - m + m')$ is valid:

$$d_{\vec{s}}(c') = c'(\vec{s}) = c(\vec{s}) - m + m' = m'$$

SymPC downsides

- Proof of k-bounded CPA security only for small k
- Ciphertext size
- Complexity: ($n \sim$ key size, $\nu = \deg(g_i) \leq |\mathbb{F}|$)
  Encrypt, decrypt $O\left(n \cdot (\nu + 1)^{n+1}\right)$ operations in $\mathbb{F}$
  Add $O\left((\nu + 1)^{n}\right)$, multiply $O\left((\nu + 1)^{2n}\right)$ operations in $\mathbb{F}$

Sparse SymPC:

- Choose sparse polynomials in encryption
  (limit the number of non-zero coefficients)
- Ciphertext size grows with multiplication

Conclusions

- Proposed a new fully homomorphic cryptosystem SymPC
- Upgraded symmetric version of Polly Cracker
- Utilized Gröbner basis in the construction
- Proved security in the information theoretical settings

Thank you for your attention!