

Key Recovery Attack on Interactable Keyed Functions

with application to a self-synchronizing stream cipher

Shahram Khazaei and Willi Meier

EPFL and FHNW
Switzerland

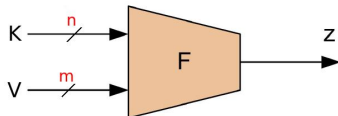
Symmetric Cryptography Seminar
Schloss Dagstuhl– Leibniz-Center for Informatics
Wadern, Germany, 15 January 2009

Outline

- ▶ Problem formalization.
- ▶ Modeling the main keyed primitives as **interactable** keyed functions $z = F(K, V)$ where the attacker can choose the public parameter V .
- ▶ Modeling a Self-Synchronizing Stream Ciphers (**SSSC**).
- ▶ Key recovery attacks on interactable keyed functions.
- ▶ Application to Klimov-Shamir T-function based SSSC.

Problem formalization

We have an **interactable** keyed Boolean function $z = F(K, V)$.



Threat model: Adversary sends public variable V of his choice to the oracle and gets back z according to a fixed unknown secret key K chosen by the oracle.

Goal: Efficient **key-recovery** attacks.

To attack a keyed primitive we need two steps:

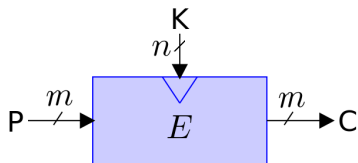
- ▶ Deriving an interactable function $F(K, V)$.
- ▶ Mounting attacks on such interactable functions.

The Main Keyed Primitives

- ▶ Block ciphers.
- ▶ Synchronous Stream Ciphers.
- ▶ Message Authentication Codes (MAC) .
- ▶ Self-Synchronizing Stream Ciphers (**SSSC**).

Block Cipher

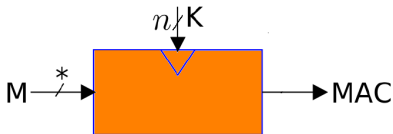
Take the function which maps the key and plaintext to one special bit of the ciphertext.



Hint: techniques like meet-in-the-middle might be useful to derive simpler functions.

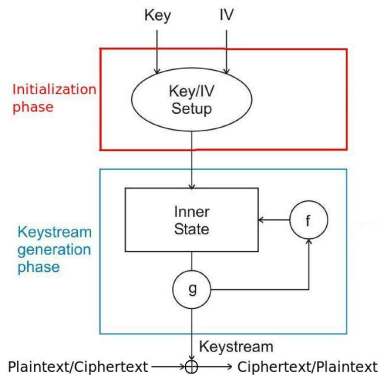
MAC

Take the function which maps the key and messages (of fixed length) to one special bit of the MAC value.



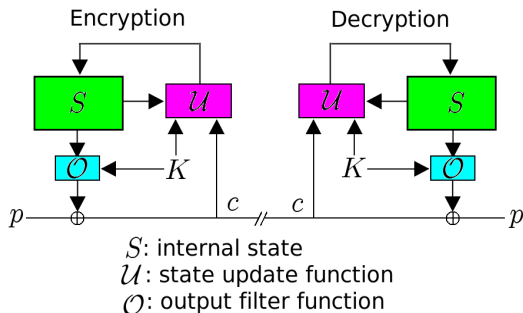
Hint: in practice one might consider the state before absorbing the last message block as unknown key and the last message block as public variable.

Synchronous Stream Cipher



Take the function which maps the key and IV to the first keystream bit.

Self-Synchronizing Stream Cipher

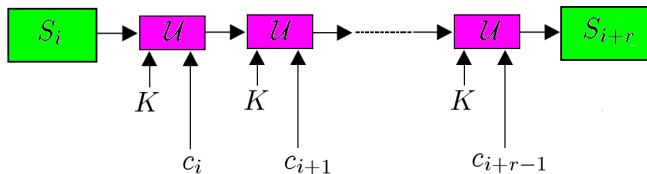


$$S_{i+1} = \mathcal{U}(c_i, K, S_i)$$

$$c_i = p_i \oplus \mathcal{O}(K, S_i)$$

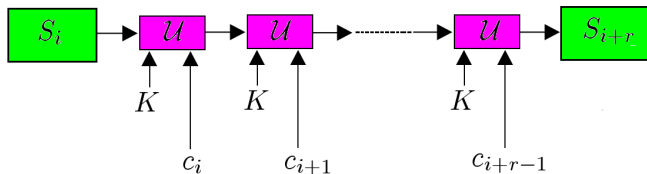
\mathcal{U} is a Self-Synchronizing Function of finite memory r , that is S_{i+r} is independent of S_i .

Self-Synchronizing Function

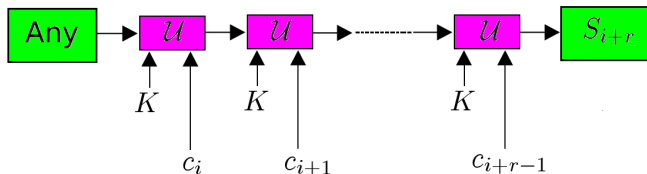


$$c_i = \hat{c}_i, \dots, c_{i+r-1} = \hat{c}_{i+r-1} \Rightarrow S_{i+r} = \hat{S}_{i+r}$$

Self-Synchronizing Function

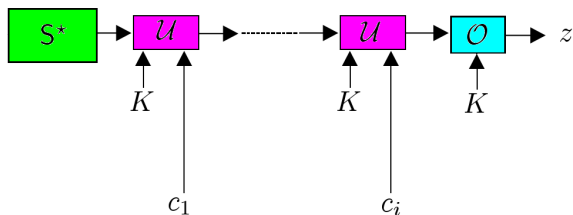


$$c_i = \hat{c}_i, \dots, c_{i+r-1} = \hat{c}_{i+r-1} \Rightarrow S_{i+r} = \hat{S}_{i+r}$$



We use this property to force the internal state to get stuck in an unknown fixed value S^* .

Attacking SSSC and Deriving Interactable Functions



Take $V = (c_1, \dots, c_i)$ as public variable.

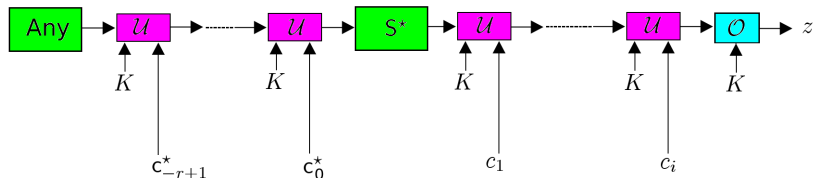
Take $K_e = (S^*, K)$ as an extended key.

Consider the interactable function $z = F(K_e, V)$.

(i is smaller than the resynchronization memory of U)

Attacking SSSC and Deriving Interactable Functions

$z = F(K_e, V)$ can be evaluated with the help of the decryption oracle in a chosen ciphertext attack scenario.



We first choose arbitrary fixed values for c_{-r+1}^*, \dots, c_0^* and c_{i+1}^* . To evaluate $F(K_e, V)$ at $V = (c_1, \dots, c_i)$:

- ▶ Send $(c_{-r+1}^*, \dots, c_0^*, c_1, \dots, c_i, c_{i+1}^*)$ to the decryption oracle.
- ▶ Receive the decrypted plaintext $(p_{-r+1}, \dots, p_{-1}, p_0, \dots, p_{i+1})$.
- ▶ Set $F(K_e, V) = p_{i+1} \oplus c_{i+1}^*$.

How to attack an interactable Boolean function?

Previous Works

Distinguishing Attacks Based on Polynomial Description

Consider $F(K, V)$ with $K = (k_1, \dots, k_n)$ and $V = (v_1, \dots, v_m)$.

Idea: Choose a list of l ($\leq m$) IV bits $\{v_{i_1}, v_{i_2}, \dots, v_{i_l}\}$, consider all key bits and the remaining IV bits as parameters and focus on the resultant parametric Boolean function $g(v_{i_1}, v_{i_2}, \dots, v_{i_l})$.

Fact: Adversary can compute the truth table (as well as the algebraic normal form (ANF)) of g with 2^l queries by dealing with the oracle.

Expectation: Each monomial must appear with probability $1/2$ and independent from others in the ANF of g .

[Filiol'02],[Saarinen'06],[O'Neil'07],[Englund-Johansson-Turan'07].

Maximum Degree Monomial Test

[Englund-Johansson-Turan'07]

High degree monomials in g are more likely to exhibit non-randomness: it will take many clockings before all selected IV bits meet in the same memory cell.

Maximum degree test check if the maximum degree monomial is produced by g .

The maximum degree test turned out to be a suitable measure for diffusion.

Key Recovery Attacks Based on Polynomial Description

[Fischer-Khazaei-Meier'08]

Make partition $V = (U, W)$ with $U = \{v_{i_1}, v_{i_2}, \dots, v_{i_l}\}$ as input to g . Focus on a single coefficient $C = C(K, W)$ of $g(U)$, e.g. maximum degree monomial.

Fact: $C = C(K, W)$ depends on the key and remaining IV bits, and can be evaluated for every W of our choice by dealing with the oracle.

Key Point: Each coefficient *not only* must be uniform, *but also* must depend on the key bits still in a complex way.

If mixing is not complete, some key bits in $C(K, W)$ may have a **small or even no** influence ...

Neutrality Measure [Fischer-Khazaei-Meier'08]

Definition: The bias of the probability that complementing each key bit does not change the output of $C(K, W)$.

Neutrality measure is helpful to approximate $C(K, W)$ with a function which effectively depends on only t ($< n$) key bits.

This reduces the search space from 2^n to 2^t which can possibly reduce the total complexity as well.

Special Cases

[Vielhaber'07]: If $C(K, W)$ is a linear function of the key bits for some special value of W (for example $W = 0$) one bit of information is revealed about the key.

[Dinur-Shamir'08]: The recently proposed **Cube Attack** is based on Vielhaber's observation. In addition they noticed that:

If $F(K, V)$ is a random function of degree d (in both K and V), $C(K, W)$ is a linear function for $l = d - 1$. This shows the existence of an attack in time $n2^{d-1} + n^3$ (n is key size).

Klimov-Shamir's SSSC

Works with 64-bit words.

$$\begin{aligned}s'_0 &= s_0 + c \\ s'_1 &= s_1 - (c \lll 21) \\ s'_2 &= s_2 \oplus (c \lll 43)\end{aligned}$$

$$\mathcal{U}(c, K, S) = \begin{pmatrix} (((s'_1 \oplus s'_2) \vee 1) \oplus k_0)^2 \\ (((s'_2 \oplus s'_0) \vee 1) \oplus k_1)^2 \\ (((s'_0 \oplus s'_1) \vee 1) \oplus k_2)^2 \end{pmatrix}$$

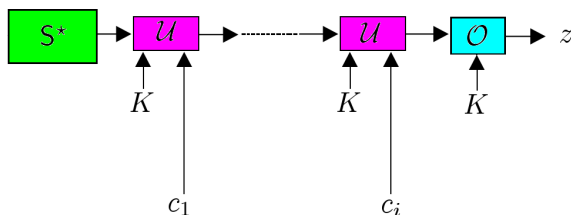
$$\mathcal{O}(K, S) = ((s_0 \oplus s_2 \oplus k_3) \lll 32) \times (((s_1 \oplus k_4) \lll 32) \vee 1)$$

ω -bit version ($\omega = 8, 16, 32, 64$): choose the rotation values to be $\lfloor \frac{\omega}{3} \rfloor$ and $\lfloor \frac{2\omega}{3} \rfloor$ ($\lfloor x \rfloor$ is the closest integer to x).

Note: the resynchronization memory is $r = \omega - 1$.

Experimental Results

We construct the family of interactable Boolean functions $\mathcal{F}_{i,j}(K_e, V)$, ($i = 1, \dots, \omega - 1$ and $j = 0, \dots, \omega - 1$), which maps $V = (c_1, \dots, c_i)$ and $K_e = (S^*, K)$ to the j -th LSB of z .



For a given i, j (or $\mathcal{F}_{i,j}(K_e, V)$) we find a subset U of l public variable bits and compute $\Gamma_{i,j}^U$, the coefficient of the product of all public variables of U .

Experimental Results: striking relations

There is always a choice for i, j and U such that Γ only depends on three key bits $k_{0,0}$, $k_{1,0}$ and $k_{2,0}$.

ω	i	j	U	$\Gamma_{i,j}^U$
8	2	0	{2}	$1 + k_{0,0}k_{1,0} + k_{0,0}k_{2,0} + k_{1,0}k_{2,0}$
16	3	0	{5}	$1 + k_{0,0}k_{1,0} + k_{2,0} + k_{0,0}k_{1,0}k_{2,0}$
16	3	0	{10}	$1 + k_{0,0} + k_{1,0}k_{2,0} + k_{0,0}k_{1,0}k_{2,0}$
32	5	0	{11}	$1 + k_{0,0}k_{1,0} + k_{2,0} + k_{0,0}k_{1,0}k_{2,0}$
32	16	0	{96, 97, 98}	$1 + k_{0,0} + k_{2,0} + k_{0,0}k_{2,0}$
64	11	0	{21}	$1 + k_{0,0}k_{1,0} + k_{2,0} + k_{0,0}k_{1,0}k_{2,0}$
64	11	0	{42}	$1 + k_{0,0} + k_{1,0}k_{2,0} + k_{0,0}k_{1,0}k_{2,0}$
64	12	0	{20}	$1 + k_{0,0}k_{1,0} + k_{0,0}k_{2,0} + k_{1,0}k_{2,0}$

Experimental Results: striking relations

There is always a choice for i, j and U such that Γ only depends on three key bits $k_{0,0}$, $k_{1,0}$ and $k_{2,0}$.

ω	i	j	U	$\Gamma_{i,j}^U$
8	2	0	{2}	$1 + k_{0,0}k_{1,0} + k_{0,0}k_{2,0} + k_{1,0}k_{2,0}$
16	3	0	{5}	$1 + k_{0,0}k_{1,0} + k_{2,0} + k_{0,0}k_{1,0}k_{2,0}$
16	3	0	{10}	$1 + k_{0,0} + k_{1,0}k_{2,0} + k_{0,0}k_{1,0}k_{2,0}$
32	5	0	{11}	$1 + k_{0,0}k_{1,0} + k_{2,0} + k_{0,0}k_{1,0}k_{2,0}$
32	16	0	{96, 97, 98}	$1 + k_{0,0} + k_{2,0} + k_{0,0}k_{2,0}$
64	11	0	{21}	$1 + k_{0,0}k_{1,0} + k_{2,0} + k_{0,0}k_{1,0}k_{2,0}$
64	11	0	{42}	$1 + k_{0,0} + k_{1,0}k_{2,0} + k_{0,0}k_{1,0}k_{2,0}$
64	12	0	{20}	$1 + k_{0,0}k_{1,0} + k_{0,0}k_{2,0} + k_{1,0}k_{2,0}$

For $\omega = 64$, the three relations give 1.75 bits of information about the key.

Experimental Results on 64-bit version

$\Gamma_{i,j}^U$ depends on t_w bits of the ciphertext and $t_k = t'_k + t_s$ bits of the extended key (t'_k bit of original key and t_s bits of the state).

ω	i	j	U	t_k	t_w	t'_k	t_s	comment
64	1	0	\emptyset	160	64	65	95	
64	1	0	$\{u\}$	90	51	30	60	$32 \leq u \leq 42$
64	31	0	$\{64\}$	89	1274	89	0	
64	31	0	$\{130\}$	88	1243	88	0	
64	31	0	$\{129, 130\}$	84	1158	84	0	
64	31	0	$\{150, 151\}$	84	1155	84	0	

Experimental Results on 64-bit version

Finding weak ciphertext bits in a systematic way for $\Gamma_{1,0}$

U	t_k	t_w	t'_k	t_s	Time
$\{41\}$	90	51	30	60	$2^{97.5}$
$\{9, 41\}$	87	50	29	58	$2^{95.4}$
$\{8, 9, 41\}$	84	49	28	56	$2^{93.4}$
$\{7 - 9, 41\}$	81	48	27	54	$2^{91.3}$
\vdots					\vdots
$\{1 - 9, 33 - 41\}$	47	34	13	34	$2^{70.5}$
$\{1 - 9, 32 - 41\}$	45	33	12	33	$2^{69.5}$

We expect that by starting from the last line and going backwards to the first line, a large amount of information about the involved $t_k = 90$ unknown bits (including $t'_k = 30$ effective key bits) is revealed for a non-negligible fraction of the keys in time $2^{69.5}$.

Conclusions

- ▶ Contributions to the recent framework of attacking interactable function as a black box.
- ▶ Deriving interactable functions for SSSC.
- ▶ Application to Klimov-Shamir's SSSC proposal.
- ▶ Providing strong key leakage for this construction.

Thank you!