PRINT Cipher

Manohar Lal Das[1], Aman Khan[2] and Aayush Deshmukh[3]

[1] IIT Bhilai, Raipur, India, manoharlal@iitbhilai.ac.in

[2] IIT Bhilai, Raipur, India, amankhan@iitbhilai.ac.in

[3] IIT Bhilai, Raipur, India, aayushd@iitbhilai.ac.in

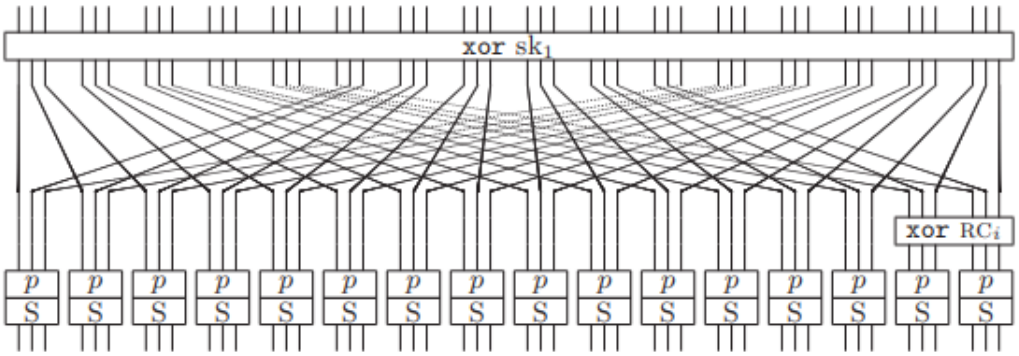
Abstract. Print Cipher is one of the lightweight SPN network with 48-bit and 96-bit block cipher for IC-printing. It is design to make use of the properties of IC-printing technology. Print cipher is still in the beginning phase of their development but allow the production of different circuits at low cost.

Keywords: SPN · IC-printing

1 Introduction

In order to identify items using smart bar-codes we use RFID tags and sensors, the security of constrained hardware environments such as RFID tags is major concern in cryptography now a days. PRINTcipher is a 48/96-bit block cipher proposed in CHES 2010 this supports the 80/160-bit secret keys. The key and attractive properties of PRINTcipher are that all rounds use the same round key and differ only by a round counter and that the linear layer is partially key-dependent. The best attack results on PRINTcipher are in-variance subspace attacks on the full $PRINTcipher-48/96$.

Most known cryptanalytic results on PRINTcipher are based on weak keys. The best attack results on PRINTcipher are invariance subspace attacks on the full $PRINTcipher-48/96$. $PRINTcipher-48$ attack applicable to 2^{52} weak keys and it requires 5 chosen plaintext with a negligible computational complexity. For $PRINTcipher-96$ the attack is applicable to 2^{102} weak keys and requires 5 chosen plaintext with a negligible computational complexity.



2 Main Result

2.1 Sbox Analysis

The sbox for the PRINT cipher is a 3-bit to 3-bit. Since input is 3-bit so for a b-bit block, the sbox is applied $\frac{b}{3}$ parallely. The current state for the sbox is a $\frac{b}{3}$ words, for each word same sbox is used and the next state is the concatenation of outputs.It is a balanced sbox and has a linear structure.The sbox is given in the following table :-

x	0	1	2	3	4	5	6	7
s[x]	0	1	3	6	7	4	5	2

2.1.1 Difference Distribution Table

The sbox has a differential branch number defined as $\min_{v,\, w \neq v} \{wt(v \oplus w) + wt(S(v) \oplus S(w))\}$ of **2**. The difference distribution table (ddt) which is generated using Sage is as follows :-

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	2	0	2	0	2	0	2
2	0	0	2	2	0	0	2	2
3	0	2	2	0	0	2	2	0
4	0	0	0	0	2	2	2	2
5	0	2	0	2	2	0	2	0
6	0	0	2	2	2	2	0	0
7	0	2	2	0	2	0	0	2

2.1.2 Linear Approximation Table

The linear branch number which is defined as $\min_{\alpha \neq, \beta, \mathrm{LAM}(\alpha,\beta)\neq 0}\{wt(\alpha) + wt(\beta)\}$ for this sbox is **2**. The linearity of this sbox is **4**. The linear approximation table generated from Sage is as follows:-

2.1.3 Additional Properties of Sbox

1. The component funcion in 3 variables in algebraic normal form of the sbox is

$$\mathbf{x0*x2 + x0 + x1*x2}$$

2. The interpolation polynomial for the sbox is

$$\mathbf{(a + 1)x^6 + (a^2 + a + 1)x^5 + (a^2 + 1)x^3}$$

3. The polynomials which satisfy the sbox is

- x0*x2 + x0 + x1 + y1

	0	1	2	3	4	5	6	7
0	4	0	0	0	0	0	0	0
1	0	-2	0	2	0	2	0	2
2	0	0	2	2	0	0	2	-2
3	0	2	-2	0	0	2	2	0
4	0	0	0	0	2	-2	2	2
5	0	2	0	2	2	0	-2	0
6	0	0	2	-2	2	2	0	0
7	0	2	2	0	-2	0	0	2

- x0*x1 + x0 + x1 + x2 + y2
- x0*y1 + x0 + x2+ y1 + y2
- x0*y2 + x1 + y1
- x1*x2 + x0 + y0
- x1*y0 + x1 + x2 + y0 + y2
- x0*y0 + x1*y1 + x2 + y2
- x1*y2 + x0 + x1 + y0
- x2*y0 + x1 + y0 + y1
- x2*y1 + x0 + y0
- x0*y0 + x2*y2 + x0 + x1 + x2 + y0 + y1
- y0*y1 + x2 + y0 + y1 + y2
- y0*y2 + x1 + y1
- y1*y2 + x0+ y0 + y1

x - input variables y - output variables

4. Maximum degree of component function - 2

5. Minimum degree of component function - 2

6. Maximal differential probability - 0.25

7. Absolute maximal linear bias - 2

8. Relative maximal linear bias - 0.25

2.2 Cryptanalysis of PRINT Cipher

We are discussing the weakness of PRINTcipher-48/96 on related-key attacks. Related keys that have different values in the part related to a key-dependent permutation. We construct t-round related key differential characteristics with a probability of 2^{-t}. By using these characteristics, we can recover the secret keys of $PRINTcipher - 48/96$. To recover the 80-bit secret key of PRINTcipher-48, our attack requires 4 related keys, 2^{47} related-key chosen plaintexts, and a computational complexity of $2^{60.62}$. In the case of PRINTcipher-96 we require 4 related keys, 2^{95} related-key chosen plaintext and a computational complexity of 2^{107}.

2.3 Construction of Related-Key Differential Characteristics on PRINTcipher

2.3.1 Steps to construct $t - round$ related-key differential characteristics on printcipher by using properties of a key-dependent permutation KP and S-box

Related-Key Properties on Key-Dependent Permutation and S-Box:- 3 -bit input value (y_0, y_1, y_2) of a key-dependent permutation of KP_l. If a 2-bit round key $sk^2_l{}_0, sk^2_l{}_1$ is equal to (0,0) or (0,1)

corresponding output value is computed as follows:

$$\text{(0,0): } (y_0, y_1, y_2) \rightarrow {KP_l}^{00}\ (y_0, y_1, y_2)$$
$$\text{(0,1): } (y_0, y_1, y_2) \rightarrow {KP_l}^{01}\ (y_1, y_0, y_2)$$

In the above relations, if y_0 is equal to y_1, each permutation outputs the same value and vice versa. That is, the following equation holds:

$$y_0 = y_1 \Leftrightarrow {KP_l}^{00}(y_0, y_1, y_2) = {KP_l}^{01}(y_0, y_1, y_2)$$

Properties of KP:-
Consider: $y_0 = y_1 \Leftrightarrow {KP_l}^{00}(y_0, y_1, y_2) = {KP_l}^{01}(y_0, y_1, y_2)$
Consider: $y_1 = y_2 \Leftrightarrow {KP_l}^{00}(y_0, y_1, y_2) = {KP_l}^{10}(y_0, y_1, y_2)$
Consider: $y_0 = y_2 \Leftrightarrow {KP_l}^{00}(y_0, y_1, y_2) = {KP_l}^{11}(y_0, y_1, y_2)$
Consider: $y_0 = y_1 = y_2 \Leftrightarrow {KP_l}^{01}(y_0, y_1, y_2) = {KP_l}^{10}(y_0, y_1, y_2)$
Consider: $y_0 = y_1 = y_2 \Leftrightarrow {KP_l}^{01}(y_0, y_1, y_2) = {KP_l}^{11}(y_0, y_1, y_2)$
Consider: $y_0 = y_1 = y_2 \Leftrightarrow {KP_l}^{10}(y_0, y_1, y_2) = {KP_l}^{11}(y_0, y_1, y_2)$

2.3.2 Related-Key Differential Characteristics on PRINTcipher-48

We will apply few property on proposed attack by considering related key-pair:
$(K = (SK^1, SK^2, K^* = (SK^{1^*}, SK^{2^*})$
$l = 0, ..., 15$

Case 1 (l) $SK^1 = SK^{1^*}$;	**Case 2 (l)** $SK^1 = SK^{1^*}$;	**Case 3 (l)** $SK^1 = SK^{1^*}$;
$SK^2_l = (0,0), SK^{2^*}_l = (0,1)$;	$SK^2_l = (0,0), SK^{2^*}_l = (1,0)$;	$SK^2_l = (0,0), SK^{2^*}_l = (1,1)$;
$SK^2_i = SK^{2^*}_i$where i != l;	$SK^2_i = SK^{2^*}_i$where i != l;	$SK^2_i = SK^{2^*}_i$where i != l;

Let input difference of the target round is zero. If key-pair K, k^* satisfies Case 1 (0), KP_0 has a nonzero related-key difference. We can construct 1-round related-key differential characteristic $0 \rightarrow Case1(0)$ 0 with a probability of 2^{-1}under case 1. Since PRINTcipher-48 uses the same round key for all rounds, we can extend this result to a t-round related-key differential characteristic.

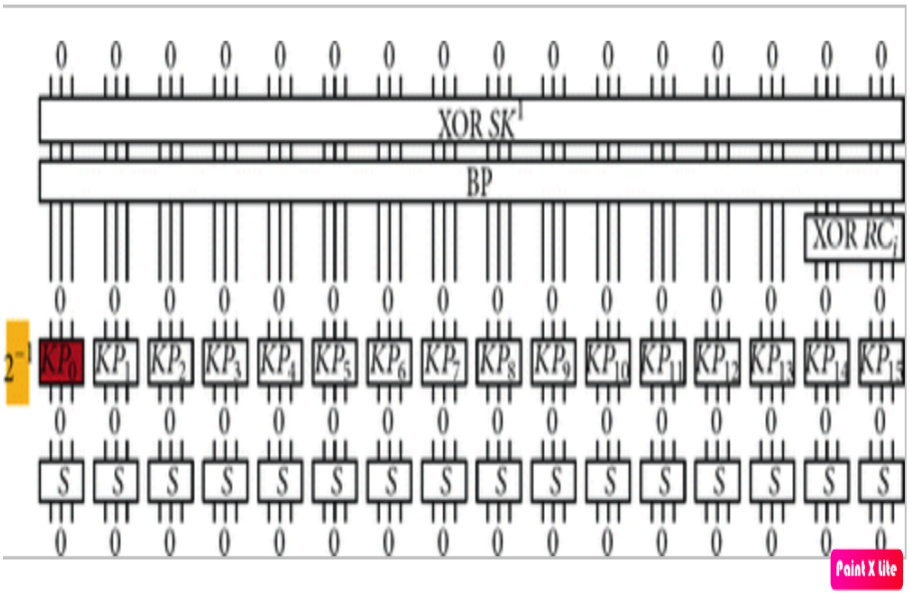


Figure 1: 1-round related-key differential characteristic under Case 1 (0)

2.3.3 Related-Key Cryptanalysis on PRINTcipher-48

we can construct t-round related-key differential characteristics on PRINTcipher-48 with a probability of 2^{-t}. These related-key differential characteristics depend on the concrete key value. These related-key differential characteristics depend on the concrete key value. To solve this, we use 4 related keys (${K_0}^{(0,0)}$, ${K_0}^{(0,1)}$, ${K_0}^{(1,0)}$, ${K_0}^{(1,1)}$). In detail, two key pairs (${K_0}^{(0,0)}$, ${K_0}^{(0,1)}$) and (${K_0}^{(1,0)}$, ${K_0}^{(1,1)}$) are considered.

2.3.4 Basic Related-Key Attack on PRINTcipher-48

44-round related-key differential characteristic with probability of 2^{-44} is needed to attack full print cipher-48. Steps for attack procedure:-

- Consider plain text strucutres each of four plaintext
- Discard the wrong ciphertext pairs from the difference between ciphertexts. For the right ciphertext pair, the output difference of round 45 should be zero.
- Guess the partial secret key
- Determine related-key pairs satisfying Case 1 (0)

2.3.5 Complexities of Basic Related-Key Attack on PRINTcipher-48

For plaintext strucutre step computational complexity is 2^48 PRINTcipher-48 encryptions. Next, we discard wrong pair with ciphertext pairs servied is $2^10(= 8.2^{44}.2^{-37})$. Total computational complexity of the attack is 2^{63}.

2.4 Experiment Results

To demonstrate the efficiency of proposal they have implemented both PRINTcipher variants in VHDL and used Synopsys DesignVision 2007.12 to synthesize them using the Virtual Silicon (VST) standard cell library UMCL18G212T3, which is based on the UMC L180 0.18μm 1P6M logic process and has a typical voltage of 1.8 Volt.

Algorithm		key size	block size	cycles/ block	Throughput (@100 KHz)	Tech. [μm]	Area [GE]
			Stream Ciphers				
Trivium	[9]	80	1	1	100	0.13	2,599
Grain	[9]	80	1	1	100	0.13	1,294
			Block Ciphers				
PRESENT	[22]	80	64	547	11.7	0.18	1,075
SEA	[17]	96	96	93	103	0.13	3,758
mCrypton	[16]	96	64	13	492.3	0.13	2,681
HIGHT	[12]	128	64	34	188	0.25	3,048
AES	[6]	128	128	1,032	12.4	0.35	3,400
AES	[11]	128	128	160	80	0.13	3,100
DESXL	[15]	184	64	144	44.4	0.18	2,168
KATAN32	[2]	80	32	255	12.5	0.13	802
KATAN48	[2]	80	48	255	18.8	0.13	927
KATAN64	[2]	80	64	255	25.1	0.13	1054
KTANTAN32	[2]	80	32	255	12.5	0.13	462
KTANTAN48	[2]	80	48	255	18.8	0.13	588
KTANTAN64	[2]	80	64	255	25.1	0.13	688
PRINTcipher-48		80	48	768	6.25	0.18	402
PRINTcipher-48		80	48	48	100	0.18	503
PRINTcipher-96		160	96	3072	3.13	0.18	726
PRINTcipher-96		160	96	96	100	0.18	967

Figure 2: Hardware implementation results of some symmetric encryption algorithms.

2.5 Conclusions

In PRINTcipher they have considered the technology of IC-printing to see how it might influence the cryptography that we use.They have proposed lightweight block cipher PRINTcipher that explicitly takes advantage of this new manufacturing approach. We related-key cryptanalysis of PRINTcipher.To recover the 80-bit secret key of PRINTcipher-48,related-key differential attack require 2^47 related-key chosen plaintexts with a computational complexity of $2^{60.62}$. Further improvement can be done on the basic related-key attack on the full PRINTcipher-48 by considering 43-round related-key differential characteristics $0 \rightarrow Case1(0)$ 0.

2.6 Testvectors

plaintext	key	permkey	ciphertext
4C847555C3E8	C28895BA327B	69D2CDB6	EB4AF95E7D37

Figure 3: Testvector for PRINTcipher-48 in hexadecimal notationAT

	Testvector 1	*Testvector 2*
plaintext	5A97E895A9837A50CDC2D1E1	A83BB396B49DAA6286CD7834
key	953DDBBFA9BF648FF6940846	D83F1CEF1084E8131AA14510
permkey	70F22AF090356768	62C67A890D558DD0
ciphertext	45496A1283EF56AFBDDC8881	EE5A079934D98684DE165AC0
	Testvector 3	*Testvector 4*
plaintext	5CED2A5816F3C3AC351B0B4B	61D7274374499842690CA3CC
key	EC5ECFEF020442CF3EF50B8A	2F3F647A9EE6B4B5BAF0B173
permkey	68EA816CEBA0EFE5	A07CF36902B48D24
ciphertext	7F49205AF958DD440ED35D9E	3EB4830D385EA369C1C82129

Figure 4: Testvectors for PRINTcipher-96 in hexadecimal notation

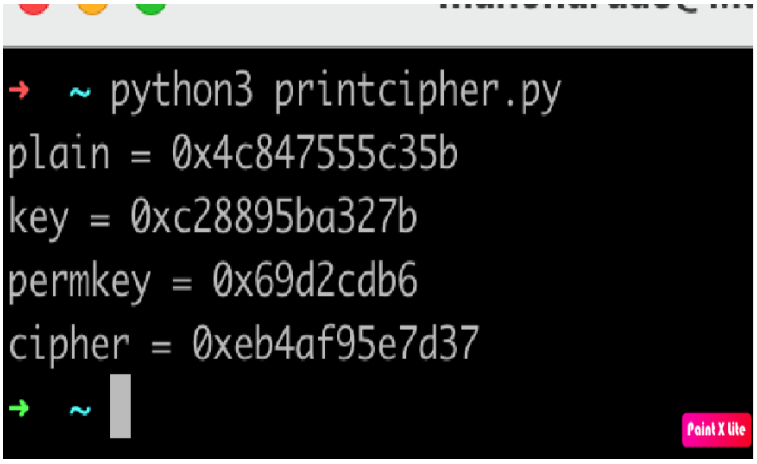


Figure 5: Sage implementation for Encryption of above plaintext

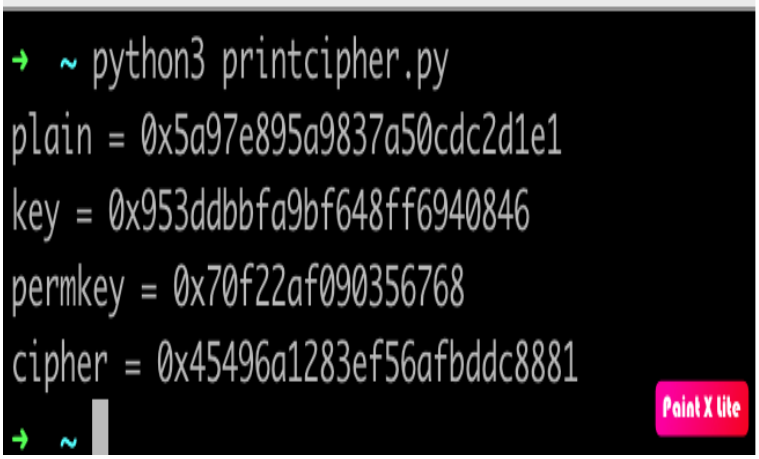


Figure 6: Sage implementation for Encryption of above plaintext

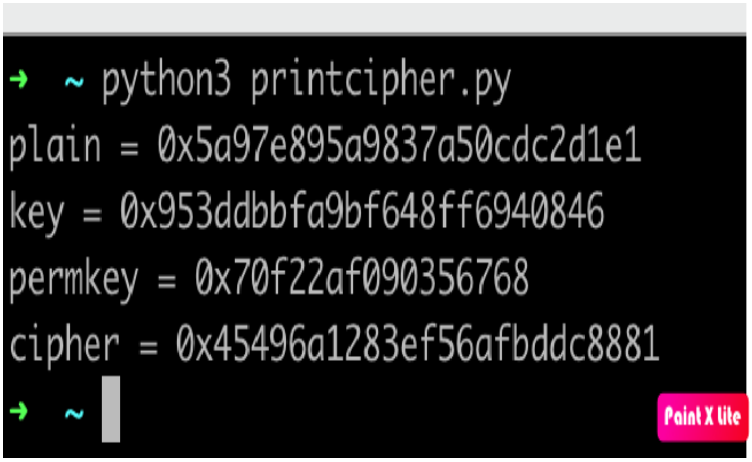


Figure 7: Sage implementation for Encryption of above plaintext

2.7 Code Snippet

```python
def enc(plaintext, long_key, short_key, block_bits = 48):
    # compute length for counter
    if block_bits == 48:
        counter = [0, 0, 0, 0, 0, 0]
    elif block_bits == 96:
        counter = [0, 0, 0, 0, 0, 0, 0, 0]
    else:
        import sys
        sys.stderr.write("ERROR: invalid block_bits\n")
        sys.exit(-1)

    text = num2bits(plaintext, block_bits)
    round_key = num2bits(long_key, block_bits)
    perm_key = num2bits(short_key, int(block_bits * 2 / 3))

    state = [None] * block_bits # temp variable
    for round_i in range(block_bits):
        # key xor
        for i in range(block_bits):
            text[i] ^= round_key[i]

        # linear diffusion
        for i in range(block_bits - 1):
            state[(3 * i) % (block_bits - 1)] = text[i]
        state[block_bits - 1] = text[block_bits - 1]

        # round counter
        counter = _update_round_counter(counter)
        for i, x in enumerate(counter):
            state[i] ^= x

        # keyed sbox
        for i in range(int(block_bits / 3)):
            before = bits2num(state[(3 * i):(3 * i + 3)])
            after = num2bits(_sbox[bits2num(perm_key[2*i : 2*i + 2])][before], 3)
            for j in range(3):
                text[3 * i + j] = after[j]

    return bits2num(text)
```

Figure 8: Encryption function of PRINTcipher