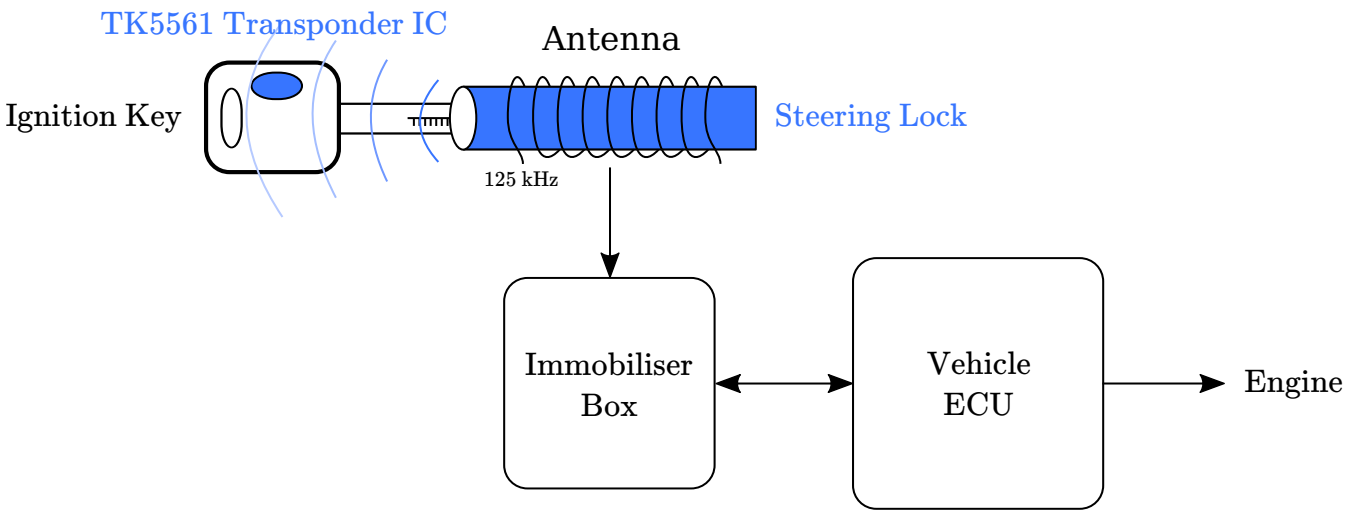
Dismantling the AUT64 Automotive Cipher

Christopher Hicks, Flavio D. Garcia, David Oswald

September 11, 2018

AUT64 immobiliser solution



- The AUT64 immobiliser solution comprises a transponder IC package (e.g. Atmel TK5561) and a vehicle immobiliser box (e.g. Mazda LC62675G2).
- (2016, RKE Garcia et. al)

Dismantling AUT64


```
aut64_2_7to1:
clra                    ; Clear bytes 50,51.
sta     byte_50
sta     byte_51
lda     aut64_roundN
asla
asla
asla
sta     byte_54         ; $54 = roundN<<3 : roundN * 8 : {0,8,16,24,...,184}
ldx     #7              ; X = 7
```

```
aut64_f:                ; $55 = loop counter
stx     byte_55
lda     $D8, x
tax                     ; X = $D8,ctr
and     #$F0 ; ''       ; A = ($D8,ctr)&0b11110000 (upper nibble)
sta     byte_52         ; $52 = ($D8,ctr)&0xf0 = *un*
aslx
aslx
aslx
aslx                    ; X = ($D8,ctr)<<4 (lower nibble<<4)
stx     byte_53         ; $53 = ($D8,ctr)<<4 *ln*
ldx     byte_54         ; X = roundN
lda     $9EA, x         ; A = table_un[roundN*8+byteN]
coma
and     #7              ; ~A&7
tax
lsrx                    ; X = [~(t_un,roundN*8+byteN)]&7>>1
lda     $E8, x          ; A = E8,X = make_crypto_out_E8,X
bcs     loc_9AD         ; Branch C set from lsrx
```

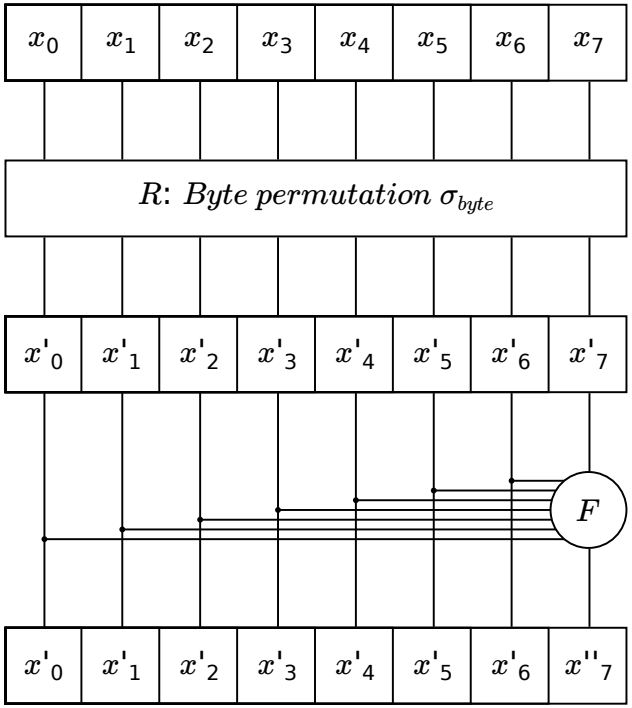
Cross-reference patents, datasheets and prior work

AUT64 Description

- A 64 bit block cipher
- 120 bit key!
- Unbalanced Feistel Network
- 8,24 rounds
- Key-dependent features and security
- Until now, no in-depth cryptanalysis or study of immobiliser implementation

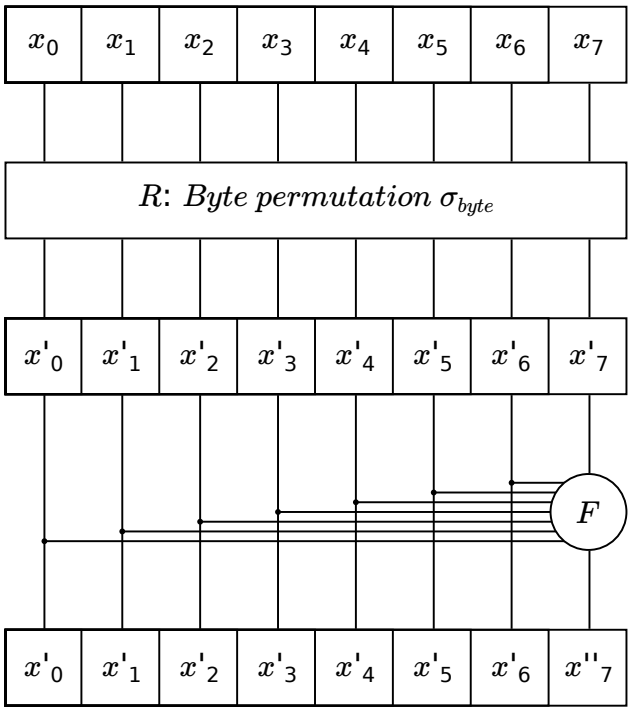
Security model

The AUT64 Block Cipher

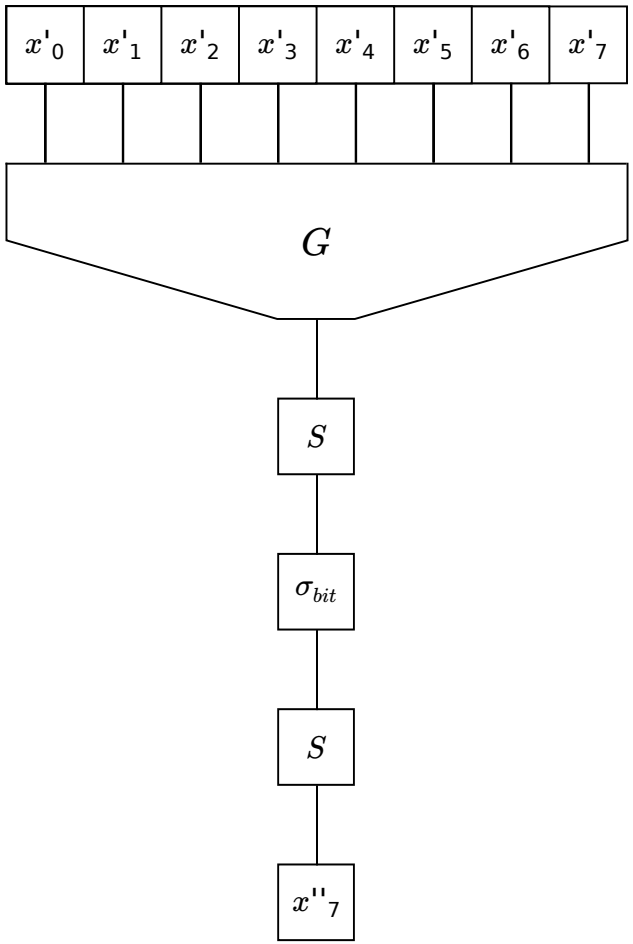


(a) One round of AUT64. The seventh byte of the state x''_7 is changed in each round.

The AUT64 Block Cipher

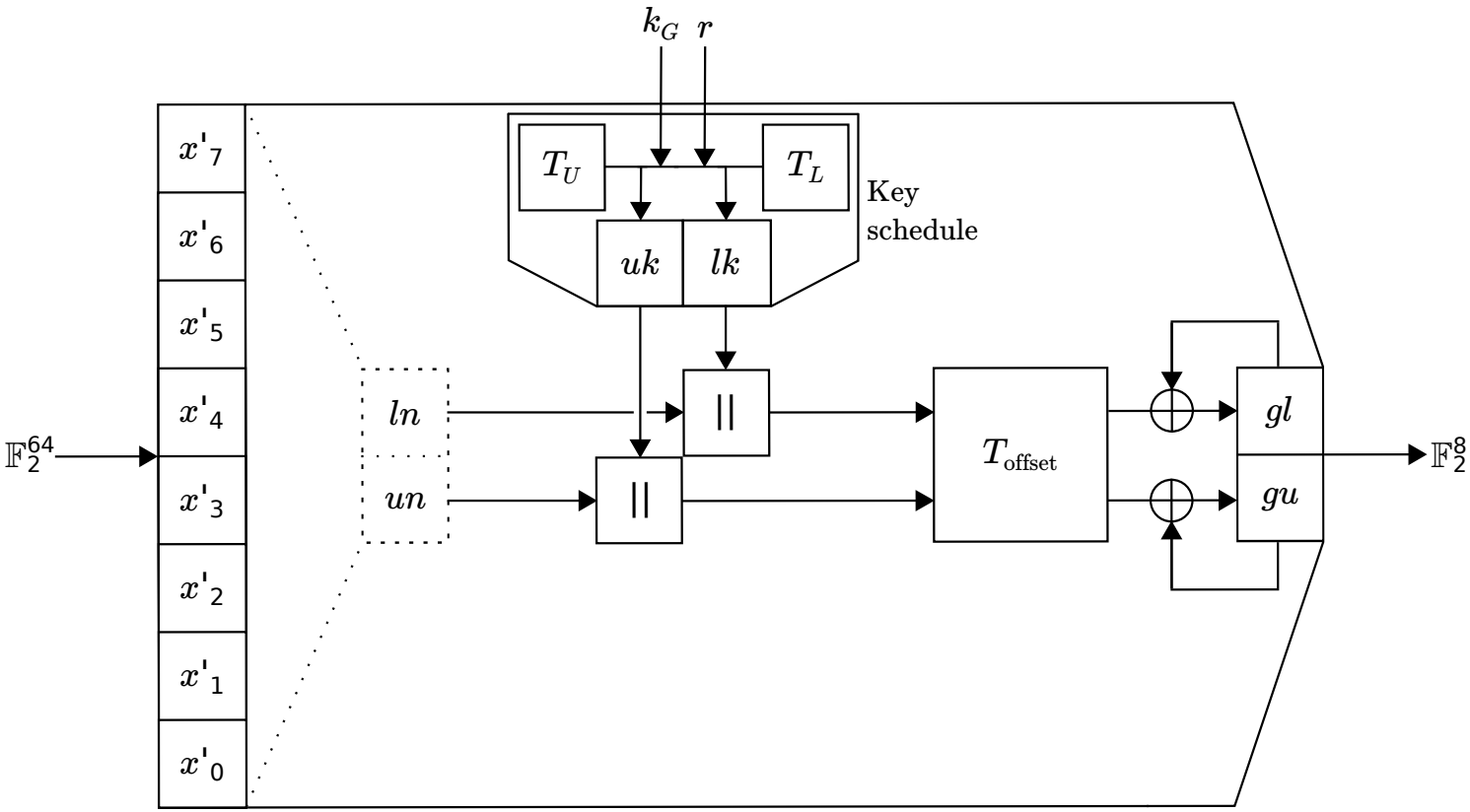


(a) One round of AUT64. The seventh byte of the state x_7'' is changed in each round.



(b) The AUT64 Feistel function F.

$G(k_G, M)$



- All operations nibble-wise
- Three look-up tables

Keys

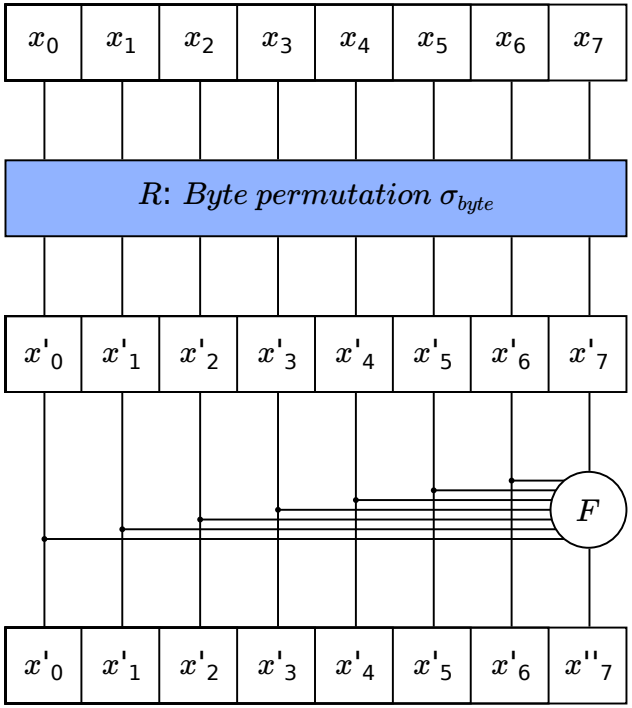
An AUT64 key is a triplet $K \in \langle k_G, k_\sigma, k_\tau \rangle$ where

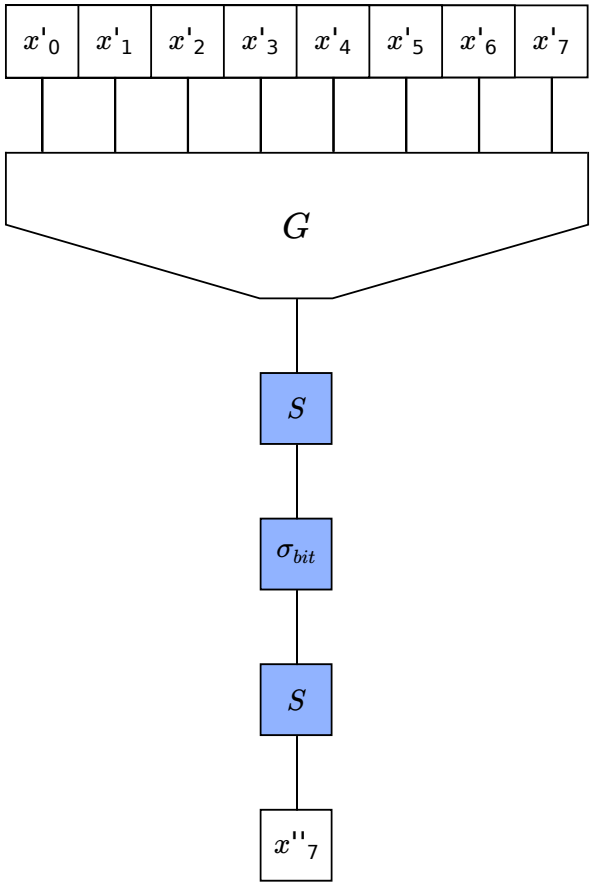
1. $k_G \in \mathbb{F}_2^{32}$;
2. k_σ defines an 8-element permutation. $8 \times 3 = 24$ bits;
3. k_τ defines a 4×4 S-Box. $16 \times 4 = 64$ bits

Nominal $32 + 24 + 64 = 120$ bit key size.

AUT64 Key Dependence

- AUT64 has a key-dependent substitution and permutations.





Key Entropy

- k_σ defines both σ_{byte} and σbit
- Total entropy is $8! \approx 15.3$ bits
- k_τ defines 4×4 S-box S
- Total entropy is $16! \approx 44.25$ bits

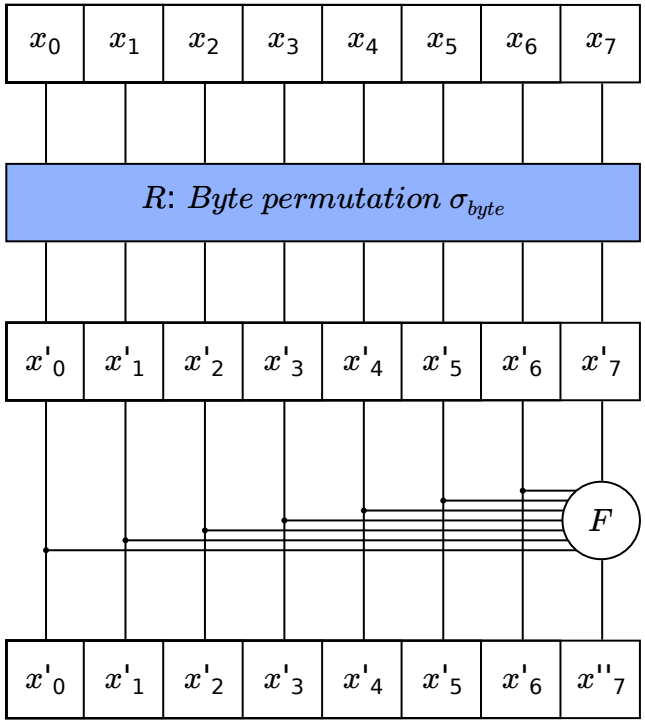
Around 91.55 bits of key entropy.

Key Entropy

Around 91.55 bits of key entropy.

- σ_{byte} must be cyclic: $|k_\sigma| = 7! \approx 2^{12.3}$
- k_τ should be non-linear:
- Result by Saarinen identifies 4 classes of 'golden' 4×4 S-Boxes, if only these were chosen $|k_\sigma| \approx 2^{39.45}$.
- k_G should not contain 0 valued nibbles: $|k_G| \approx 2^{31.3}$

Perhaps only ≈ 83 bits of key entropy.



AUT64 cryptographic weaknesses

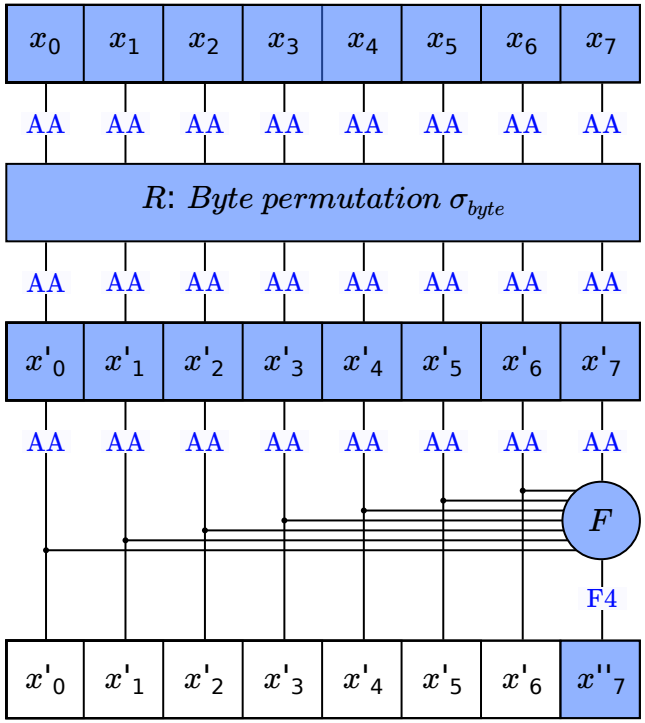
- Mostly focus on chosen-plaintext cryptanalysis.
- We try to distinguish the output of 8-round AUT64 from a random permutation.

8 round AUT64

There exists a byte in the output which was determined only by applying the round function $F(K, \sigma_{\text{byte}}(M))$.

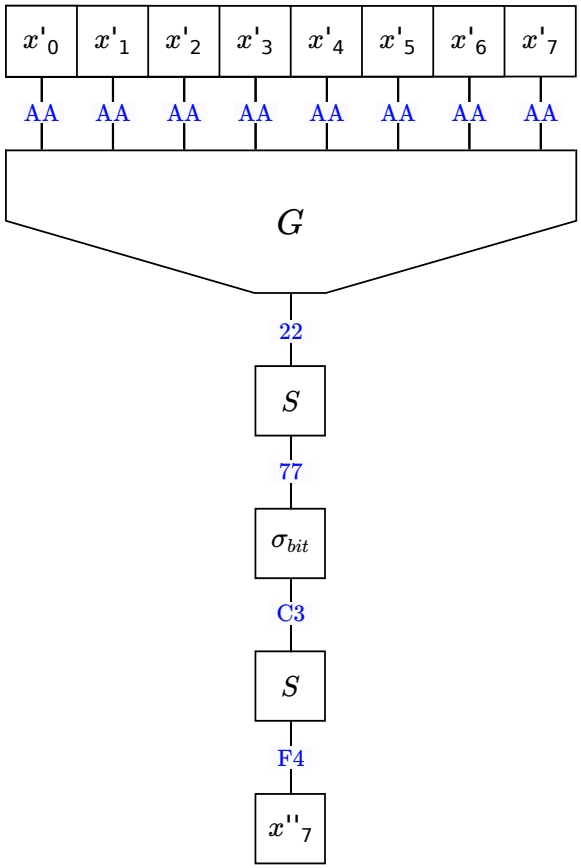
Subsequent rounds introduce increasing uncertainty.

We can learn $F(K, M)$

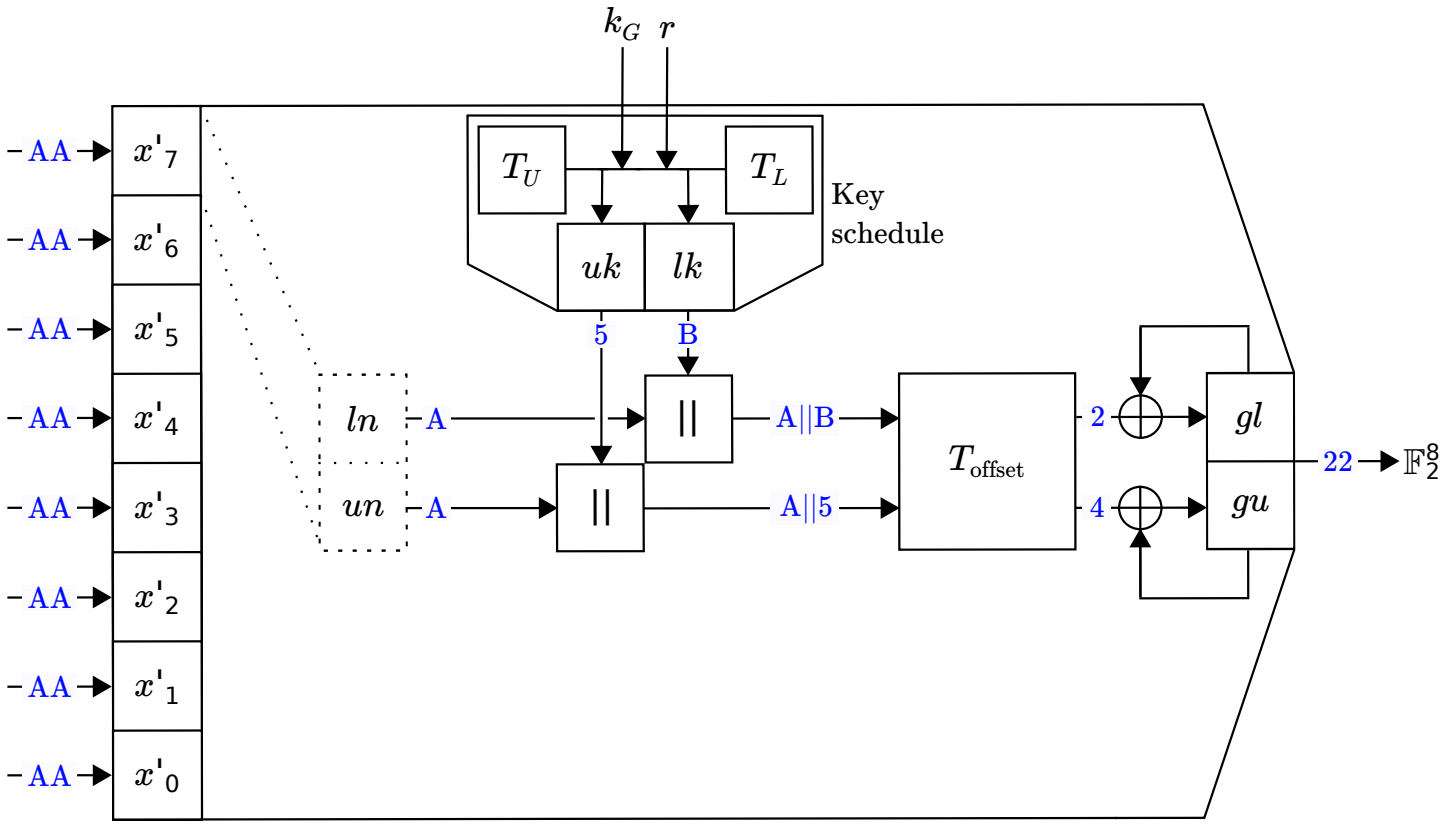


$F(K, M)$

- S is only 4×4;
- Ideally F, G would output uniformly from $[0, 1, \ldots, 255]$.
- $F(K, M) = S\Big(\sigma_{\text{bit}}\big(S(G(M))\big)\Big)$



$G(k_G, M)$



- All operations nibble-wise
- Output is not uniformly random

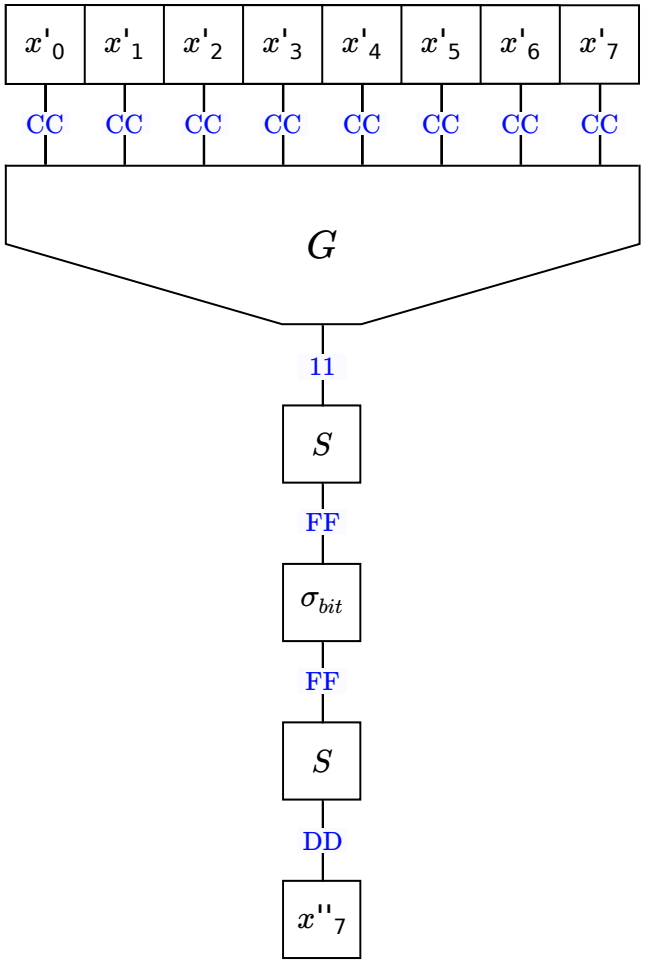
$G(k_G, M) : T_{\text{offset}}$

Key Nibble \ Input Nibble A	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	0	2	4	6	8	A	C	E	3	1	7	5	B	9	F	D
3	0	3	6	5	C	F	A	9	B	8	D	E	7	4	1	2
4	0	4	8	C	3	7	B	F	6	2	E	A	5	1	D	9
5	0	5	A	F	7	2	D	8	E	B	4	1	9	C	3	6
6	0	6	C	A	B	D	7	1	5	3	9	F	E	8	2	4
7	0	7	E	9	F	8	1	6	D	A	3	4	2	5	C	B
8	0	8	3	B	6	E	5	D	C	4	F	7	A	2	9	1
9	0	9	1	8	2	B	3	A	4	D	5	C	6	F	7	E
A	0	A	7	D	E	4	9	3	F	5	8	2	1	B	6	C
B	0	B	5	E	A	1	F	4	7	C	2	9	D	6	8	3
C	0	C	B	7	5	9	E	2	A	6	1	D	F	3	4	8
D	0	D	9	4	1	C	8	5	2	F	B	6	3	E	A	7
E	0	E	F	1	D	3	2	C	9	7	6	8	4	A	B	5
F	0	F	D	2	9	6	4	B	1	E	C	3	8	7	5	A

- For each input nibble and k_G nibble there is a unique value XORed into the output register;
- XOR is commutative: can force 'symmetric' output byte

σ_{byte} Divide and Conquer

- We can craft inputs s.t that G will output a symmetric byte
- $\mathbb{P} = \left\{ (n \parallel n)^8 : n \in \{0, \ldots, 15\} \right\}$
- σ_{bit} will typically remove symmetry
- $\frac{2}{16}$ inputs to σ_{bit} will have symmetry preserved
- Pick byte position with greatest number of symmetric bytes
- Probabilistic

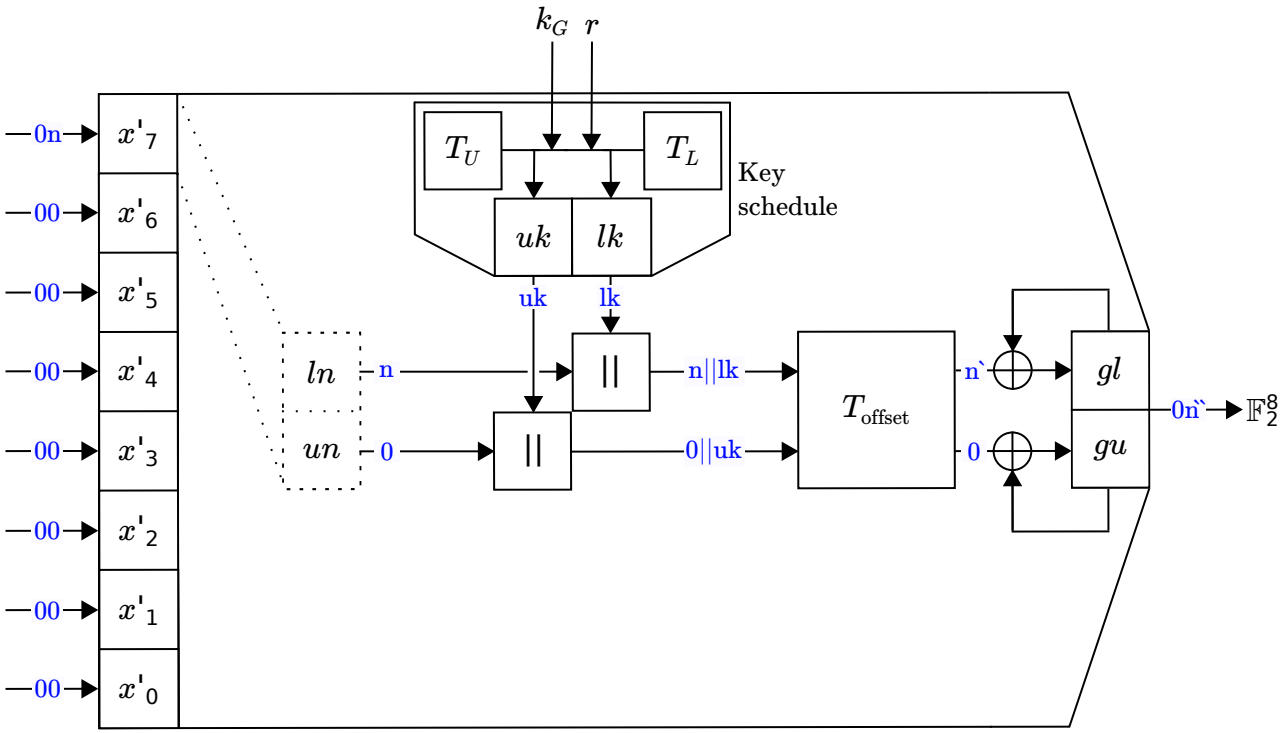


2f	fd	e7	34	22	41	77	67
21	2b	09	4e	89	e6	70	41
82	35	7b	9f	98	fd	34	1c
1a	21	a3	28	31	ba	03	1a
e7	60	73	26	aa	ab	a5	d2
a3	40	08	24	4d	8b	4a	a8
1f	55	e2	be	60	27	6d	d6
2a	b5	71	e2	06	96	8e	ce
da	5d	ee	3a	cc	f0	04	dd
89	98	f5	78	77	b5	f9	05
ab	57	e2	0b	ff	4b	9b	51
2e	d7	dc	c1	ee	84	74	34
59	f3	f8	6c	13	5a	9a	2f
3d	ed	e1	ad	d4	24	16	f6
8e	c7	ed	93	55	93	a4	13
c2	8d	52	99	bb	b2	51	82

σ_{byte} Divide and Conquer

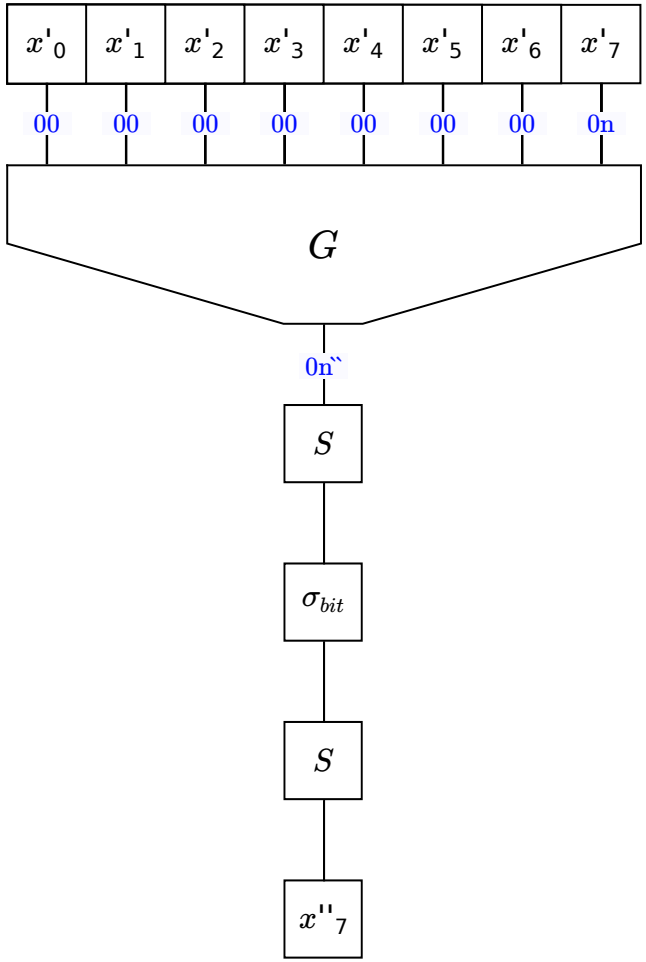
- Attack on average reduce $|k_\tau|$ from $16!$ to $2 \times 14! \approx 2^{37.3}$
- $|k_\sigma|$ is reduced from $(8-1)! \approx 2^{12.3}$ to $(8-2)! \approx 2^{8.4}$
- K entropy reduced to 76.9 bits
- Can make non-probabilistic

k_G Divide and Conquer



- We can use the known byte position of the first round r_0 to divide and conquer k_G with an attack that requires a further 16 chosen plaintexts and reduces the security to $\leq 2^{50.7}$ encryptions.

- $\mathbb{P} = \left\{ \left(n \ll (64 - 8 \times r_0) \right) : n \in \{0, \ldots, 15\} \right\}$
- For each $P \in \mathbb{P}$, try each $k_\tau, k_\sigma, k_{G_3}$.
- $|k_\tau| \approx 2^{37.3}, |k_\sigma| \approx 2^{8.4}, |k_{G_3}| = 15,$

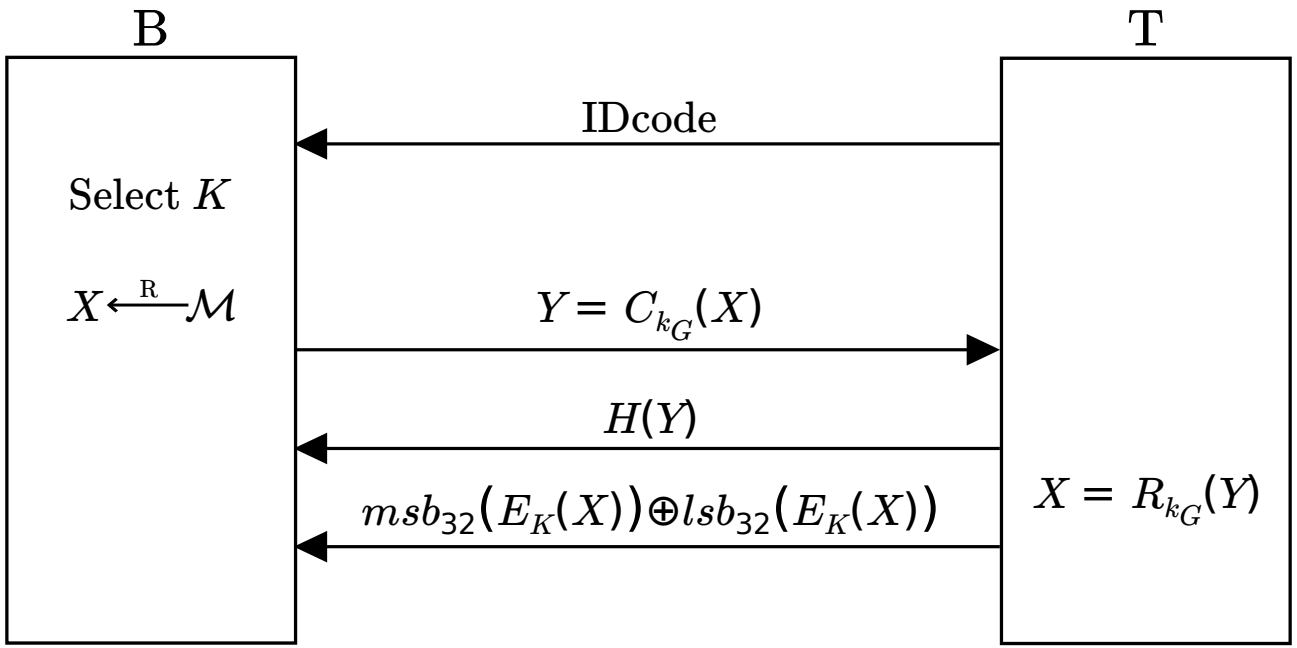


Weakness	**Keyspace (bits)**
None	120
High-level analysis	91.5
Permutation Key Size	88.5
Integral Cryptanalysis	≤ 78.8
Compression Function Divide and Conquer	≤ 50.7

AUT64 TK5561 Implementation Details

- Default 8 round implementation but 24 in the version we disassembled.
- Combined with a bespoke challenge-response protocol.
- Very weak key management:
  1. k_G is derived from transponder *IDcode*;
  2. $|k_\sigma| = 16$ per manufacturer;
  3. Some evidence k_τ also fixed.

AUT64 Challenge-Response

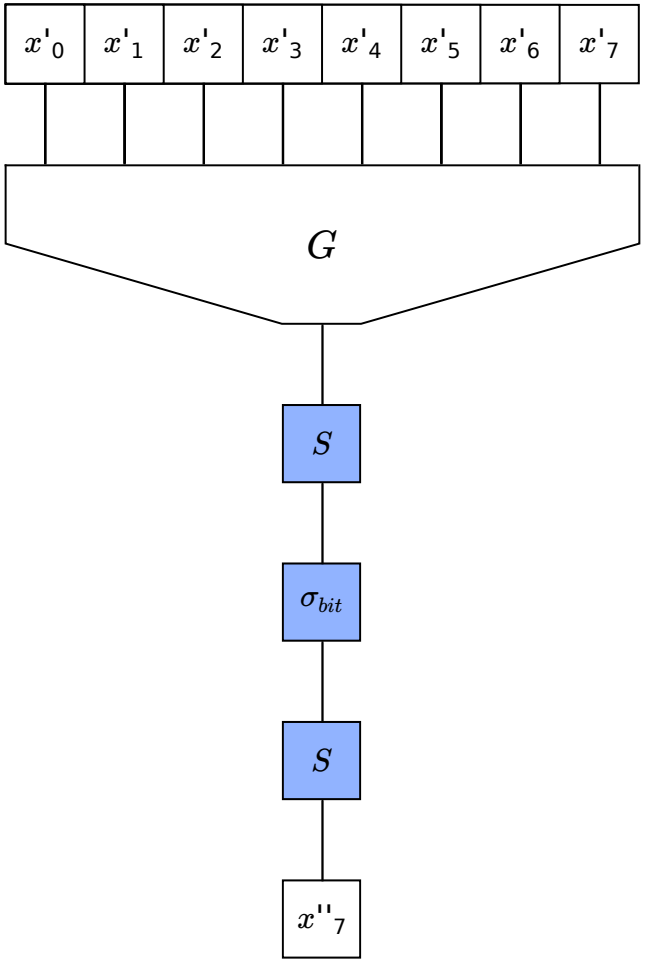


AUT64 TK5561 Implementation Details

- Default 8 round implementation but 24 in the version we disassembled.
- Combined with a bespoke challenge-response protocol.
- Very weak key management:
  1. k_G is derived from transponder *IDcode*;
  2. $|k_\sigma| = 16$ per manufacturer;
  3. Some evidence k_τ also fixed.

Results

- 8 round AUT64 is not a secure block cipher. Security less than $\approx 2^{50.7}$ encryptions despite 120 bit key.
- 8 round AUT64 with known k_G can be broken within milliseconds;
- 24 round TK5561 AUT64 is more secure, but broken in-practice owing to weak key management.

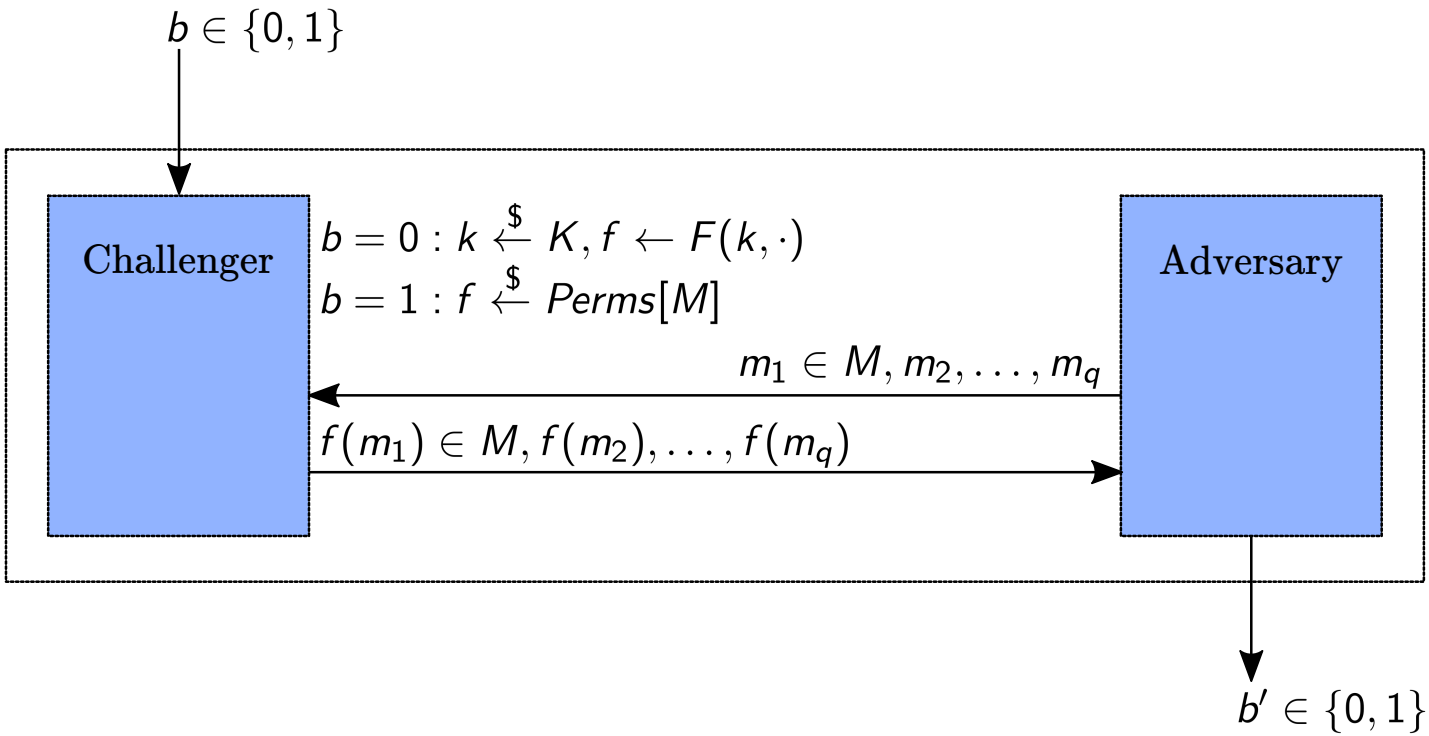

x'_0
x'_1
x'_2
x'_3
x'_4
x'_5
x'_6
x'_7
G
S
σ_{bit}
S
x''_7

Results

- 8 round AUT64 is not a secure block cipher. Security less than $\approx 2^{50.7}$ encryptions despite 120 bit key.
- 8 round AUT64 with known k_G can be broken within milliseconds;
- 24 round TK5561 AUT64 is more secure, but broken in-practice owing to weak key management.

Security Model

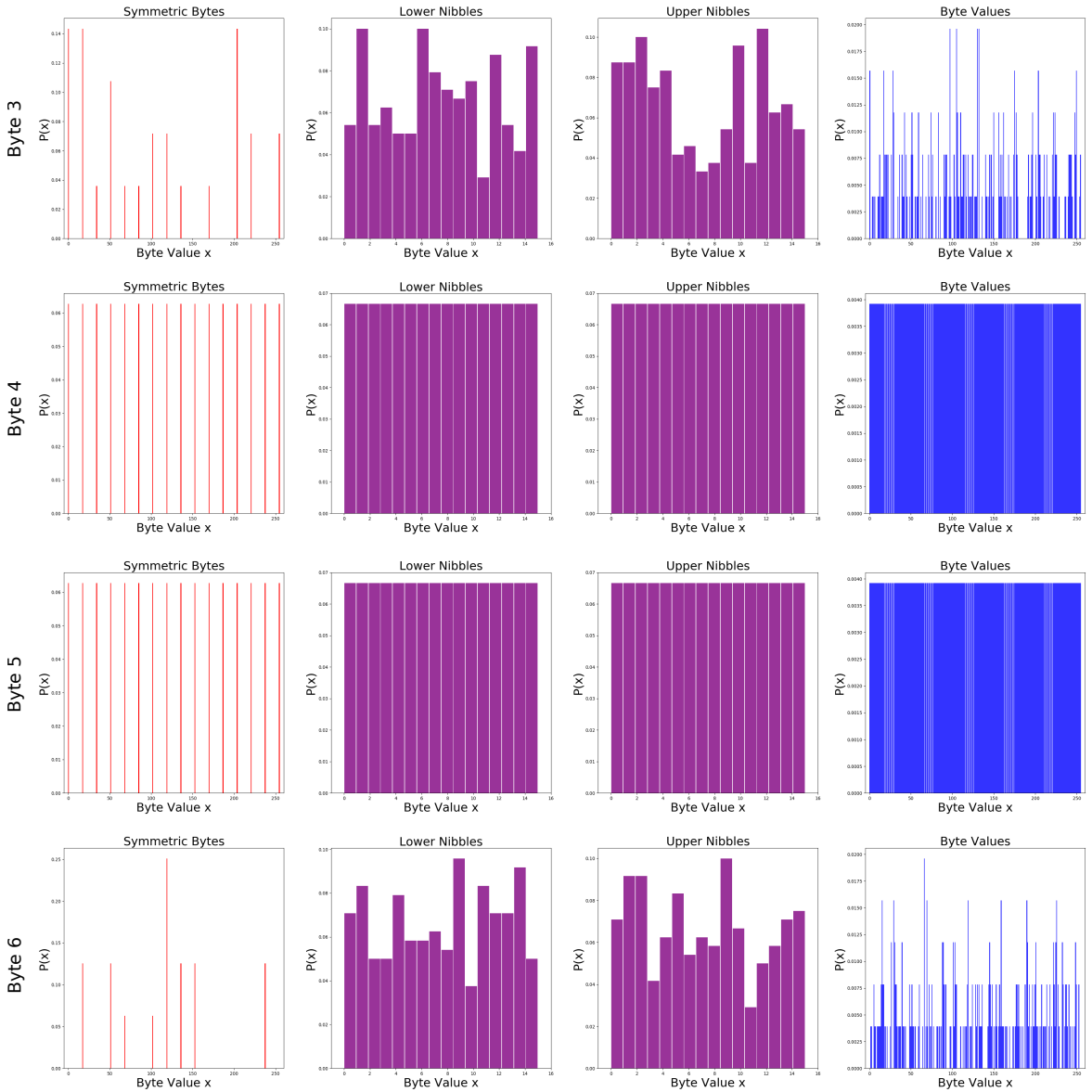
An ideal block cipher is indistinguishable from a random permutation.



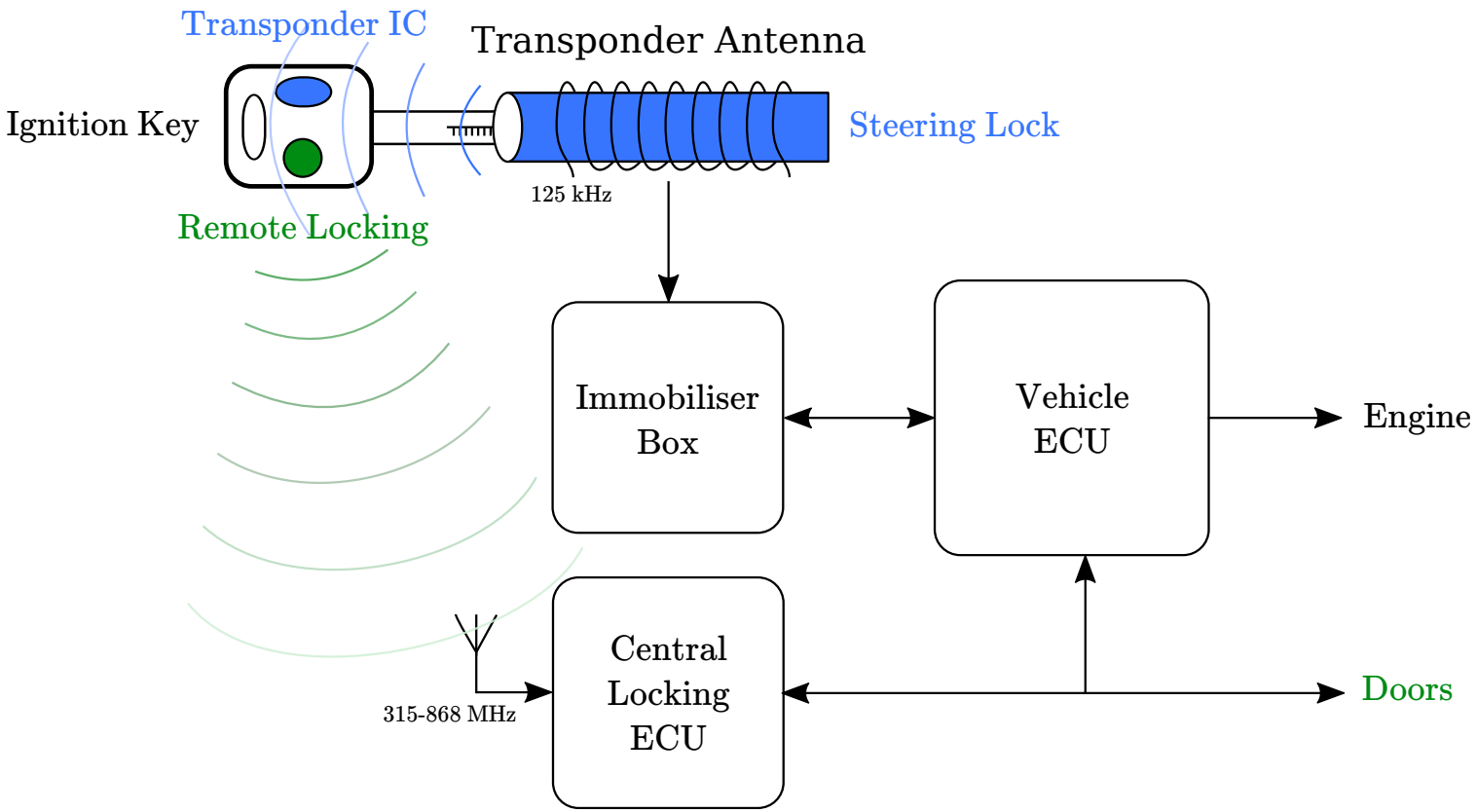
Back to High-level AUT64

Integral Cryptanalysis

- We can generalise the earlier divide and conquer attack on the first round
- $\mathbb{P} = \left\{ \left(n \parallel (\mathtt{0x00})^7 \right) : n \in \{0, \ldots, 255\} \right\}$
- Let $\mathbb{C}$ be corresponding set of ciphertexts.
- Divide $\mathbb{C}$ by byte position and then compute the XOR sum of all the bytes at each position.
- The sum is zero in the output from the first and second round.
- Can be extended beyond first two rounds.

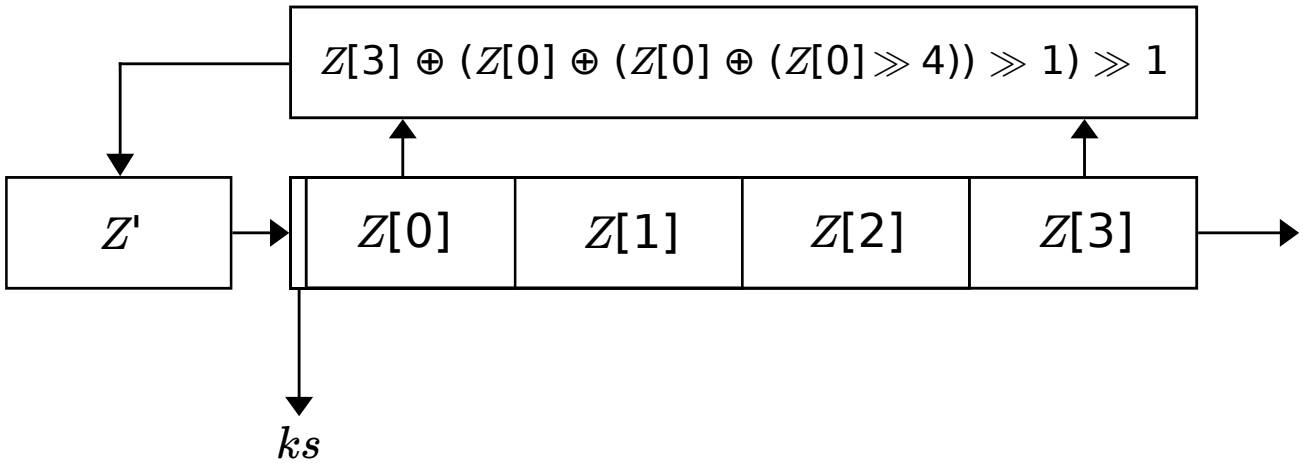

Byte 3
Byte 4
Byte 5
Byte 6
Symmetric Bytes
Lower Nibbles
Upper Nibbles
Byte Values
P(x)
Byte Value x

Automotive Anatomy



AUT64 Challenge-Response

$$Z = k_G[0] \oplus \texttt{0xD5} \parallel k_G[1] \oplus \texttt{0x89} \parallel k_G[2] \oplus \texttt{0x0C} \parallel k_G[3] \oplus \texttt{0x7B}$$



Back