Implementation Security In Cryptography

Lecture 03: Introducing Block Ciphers

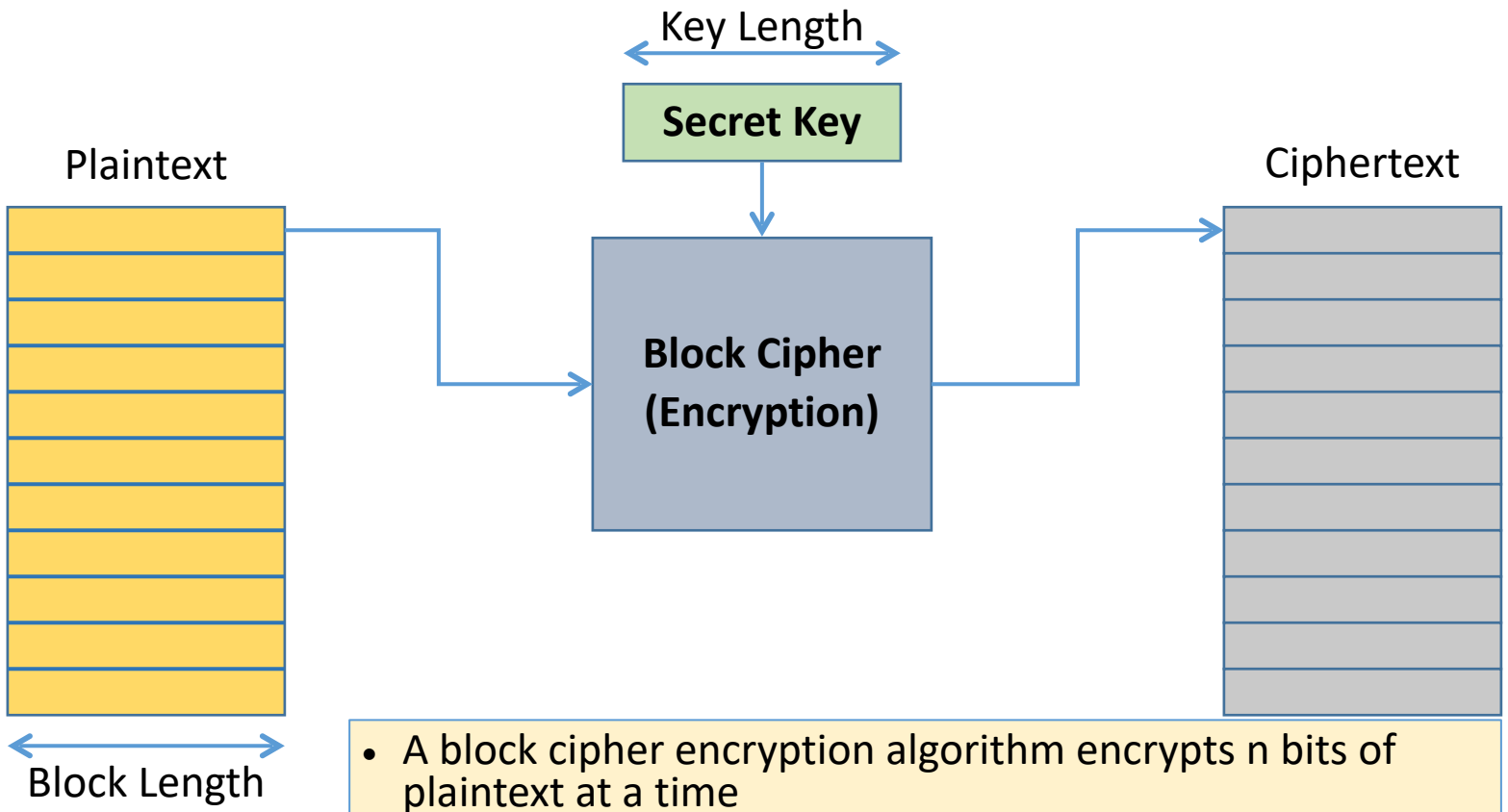
Recap

- In the last lecture
  - What people expect from crypto?
  - How they theoretically model the adversaries and the security?
  - PRG and PRFs — important building blocks for symmetric key crypto
- Public key also have similar set of security definitions with some more building blocks based on “hard problems”
- We can always “prove” if these building blocks are secure, then the crypto system is secure..
- But for this course, we are more interested in
  - How these core building blocks are implemented.
  - Are they really secure in the practical world??

Today..

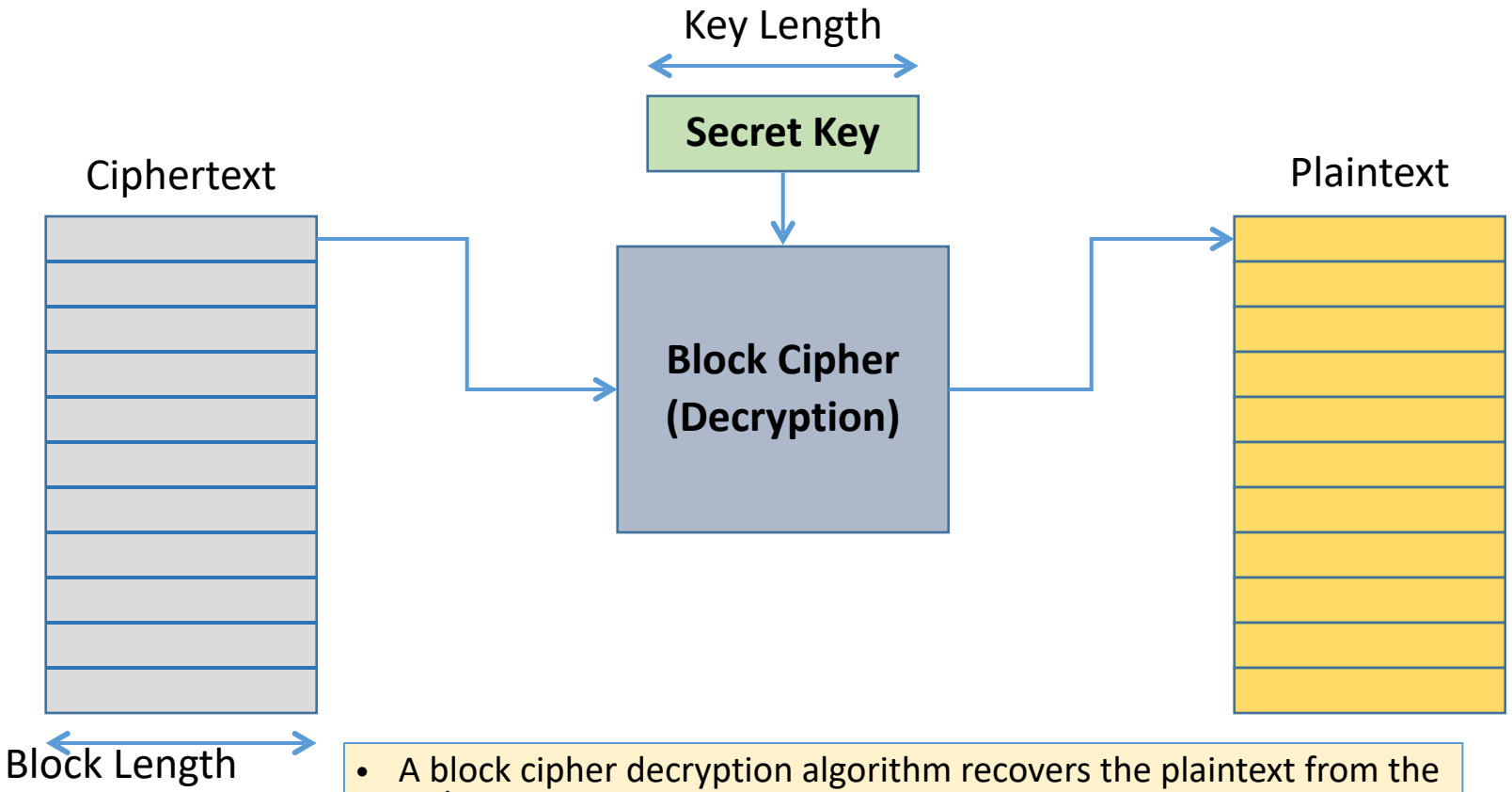
- Block ciphers — building blocks for PRFs, and sometimes PRG too..
  - PRESENT
  - AES

Block Cipher : A Symmetric Key Encryption



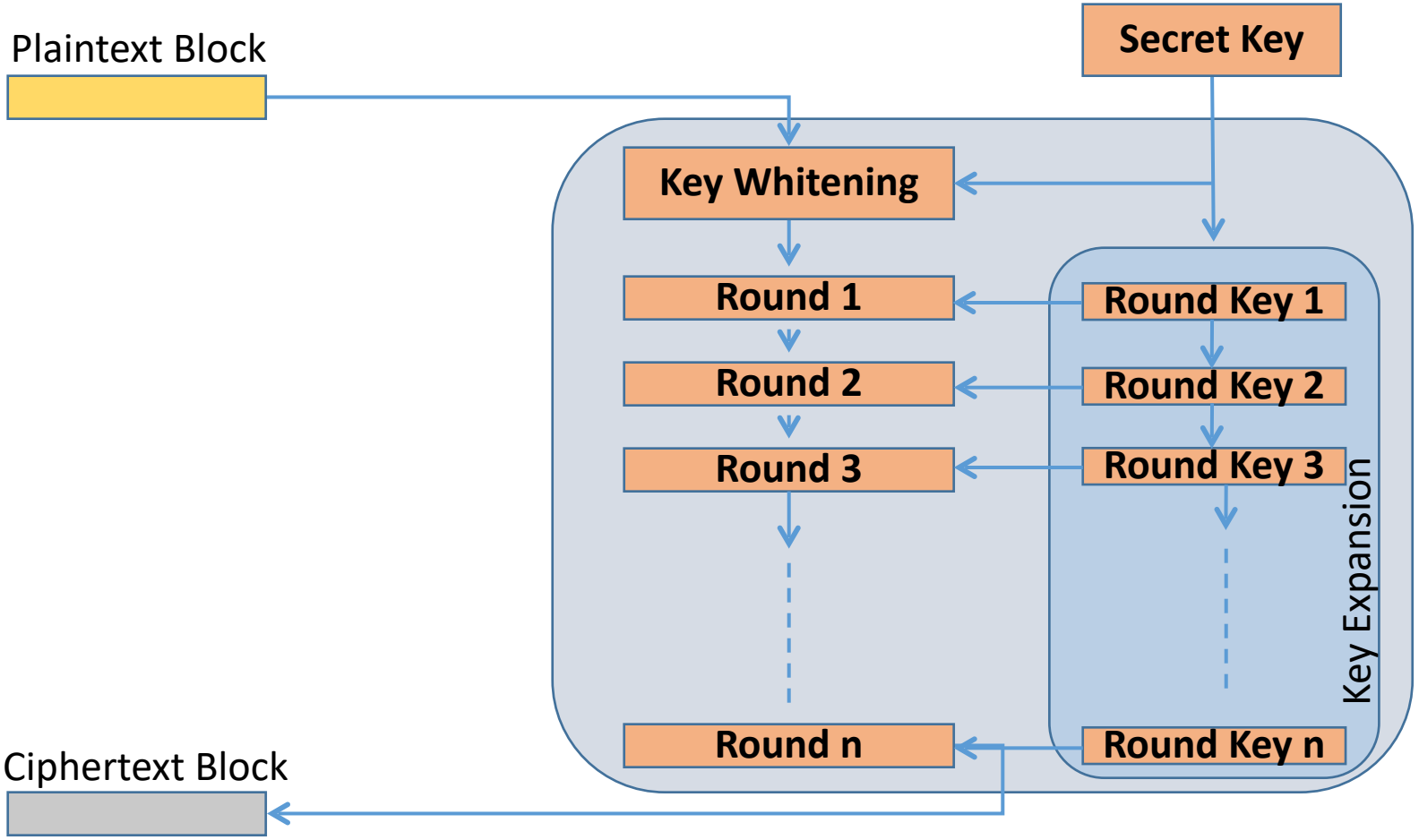
- A block cipher encryption algorithm encrypts n bits of plaintext at a time
- $C = E_k(P)$

Block Cipher : Decryption



- A block cipher decryption algorithm recovers the plaintext from the ciphertext.
- $P = D_k(C)$

Structure of a Block Cipher

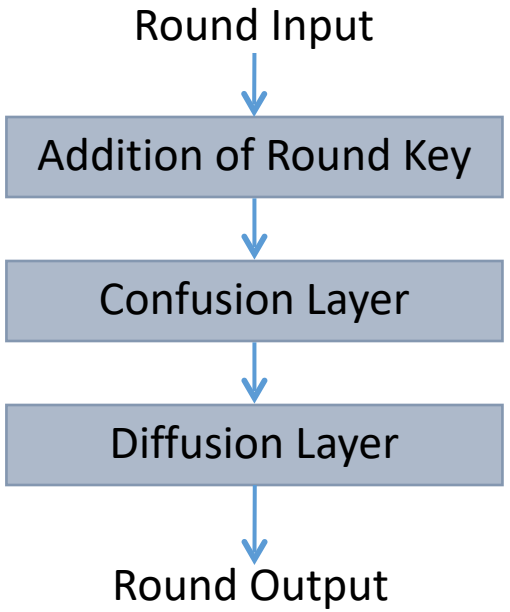


Key and Block length

- Generally 128, 192, or 256 bits
- The key and block length can be independent of each other
- The choice of length decides how effective the cipher is against brute force attacks.
  - 80 bits is considered the minimum requirement.
- Longer lengths however result in slower encryption times, thus more performance overheads.

An Encryption Round

- Consists of three operations
  - Addition of Round Key
  - ***Confusion :*** Hides the relation between the ciphertext and the key.
  - ***Diffusion :*** Hides the relation between the ciphertext and the plaintext.
- Number of rounds is determined by the cipher's resistance to known attacks.

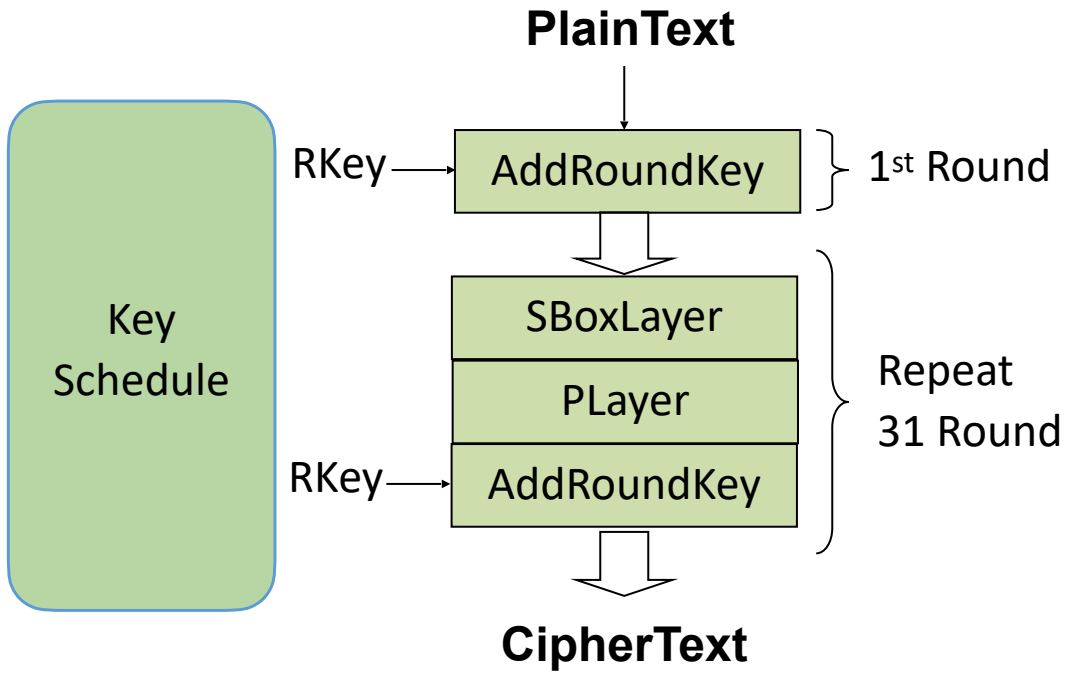


PRESENT: A Lightweight Block Cipher

- Made for lightweight applications.
- ISO standard for lightweight cryptography
- 64-bit block size
- 80/128 bit key size
- 31 rounds
- Now let us see how it is made

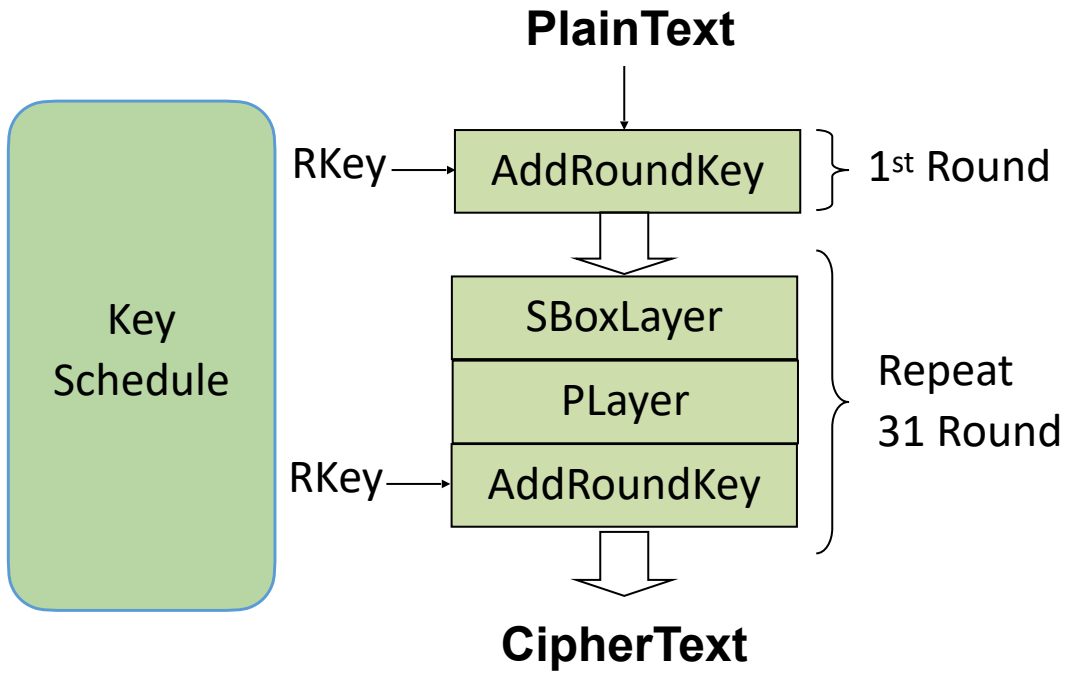
PRESENT: A Lightweight Block Cipher

- Made for lightweight applications.
- ISO standard for lightweight cryptography
- 64-bit block size
- 80/128 bit key size
- 64-bit Round keys
- Now let us see how it is made…



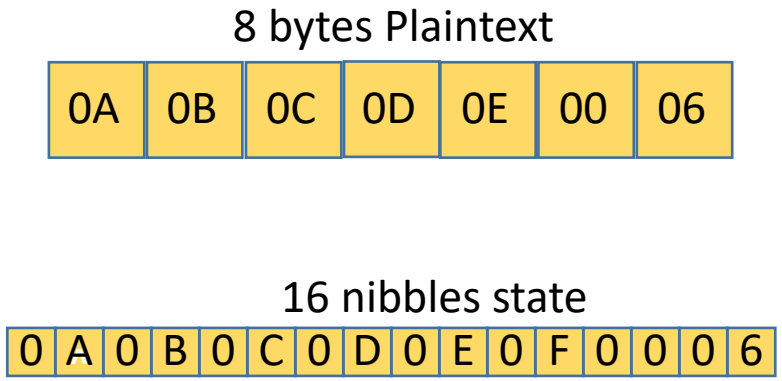
PRESENT: A Lightweight Block Cipher

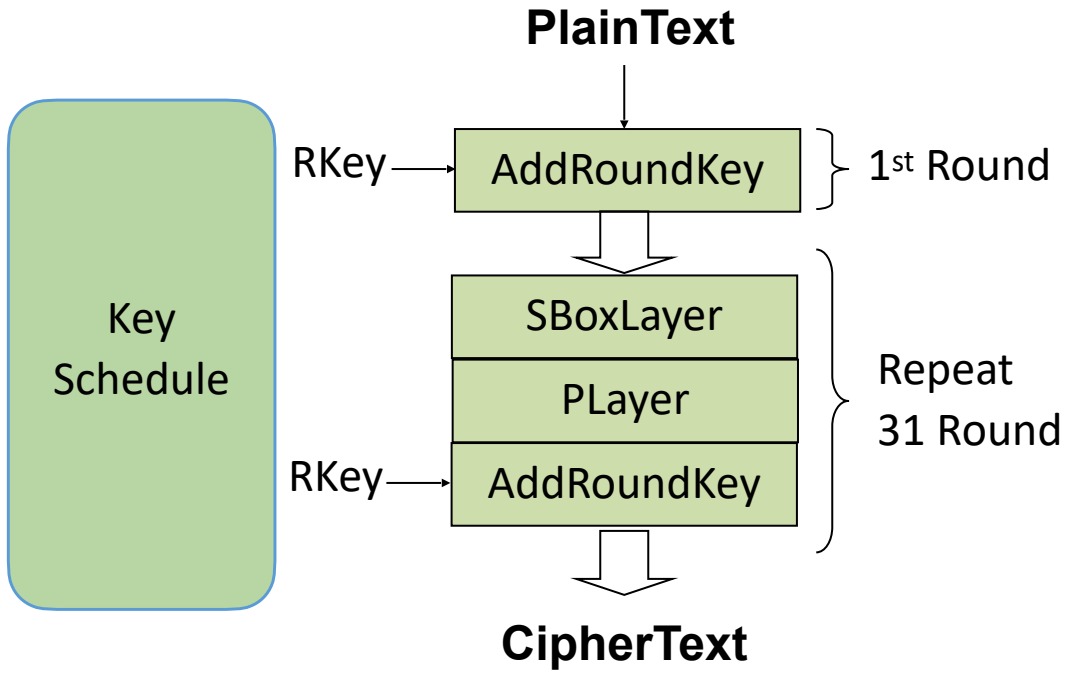
- Everything is defined as Boolean logic
- For convenience we shall use a special kind of logic representation style
  - Algebraic Normal Form (ANF)
  - The functionally complete set is (AND, XOR, 0, 1)
  - Also, a bit of notation abuse: + is $\oplus$
- Let's see a bit of ANF
  - How do we represent NOT: a + 1
  - How do we present OR: a + b + ab — see why?



PRESENT: A Lightweight Block Cipher

- The State Representation

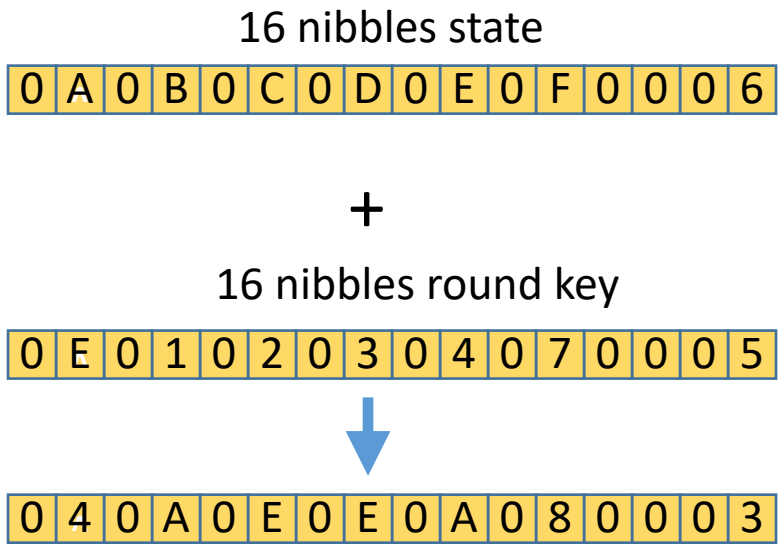


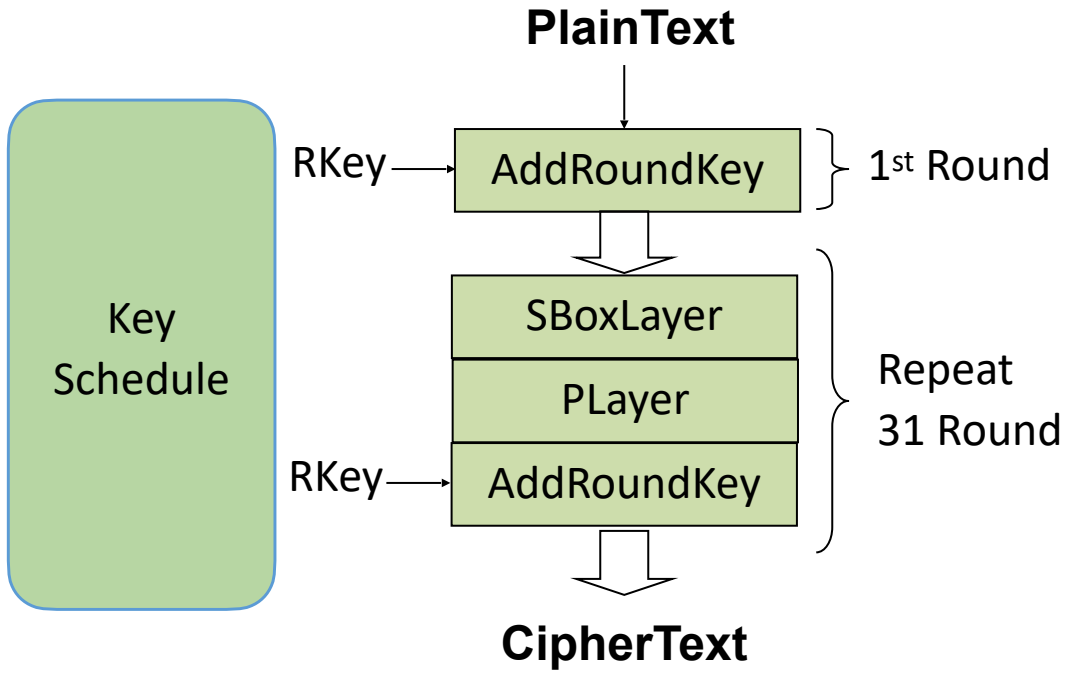


- Nibble: 4-bit chunk of the state.
- In hardware — straightforward
- In software — each 32 bit register contains 8 nibbles. So the entire state can be contained in 2 registers (embedded world).

PRESENT: A Lightweight Block Cipher

- AddRoundKey

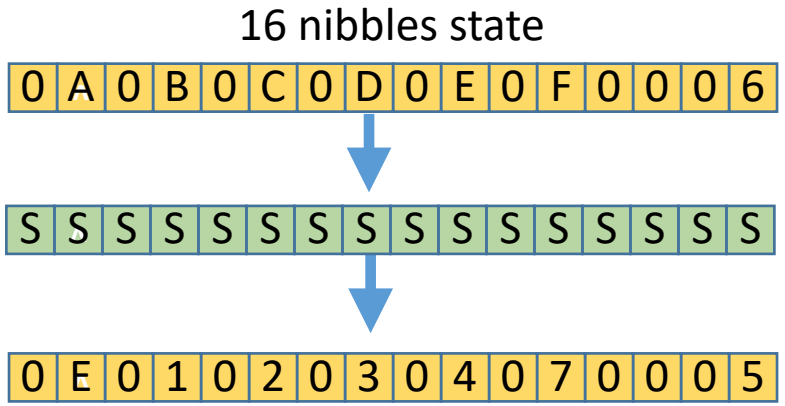


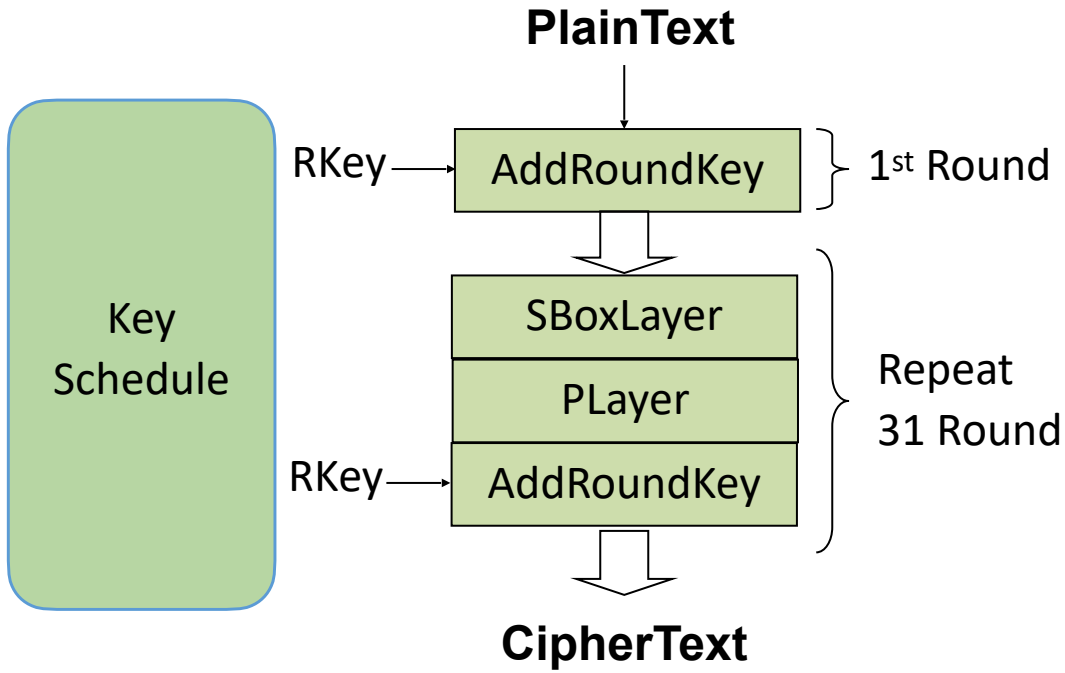


- Remember "+" means XOR
- Bitwise XOR operation
- Very easy to do in both software and hardware

PRESENT: A Lightweight Block Cipher

- SBoxLayer





x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

- S-Box is a bijection from 4-bits to 4-bits
- It is not “just” some random mapping
- It has certain properties which makes it resistant against differential and other attacks

PRESENT: A Lightweight Block Cipher

- SBoxLayer

- It has certain properties which makes it resistant against differential and other attacks
  - Firstly, it is a balanced function
  - Secondly, one bit change in input at least propagate to 2 output bits of the S-Box
  - There are many more to prevent against differential and linear attacks

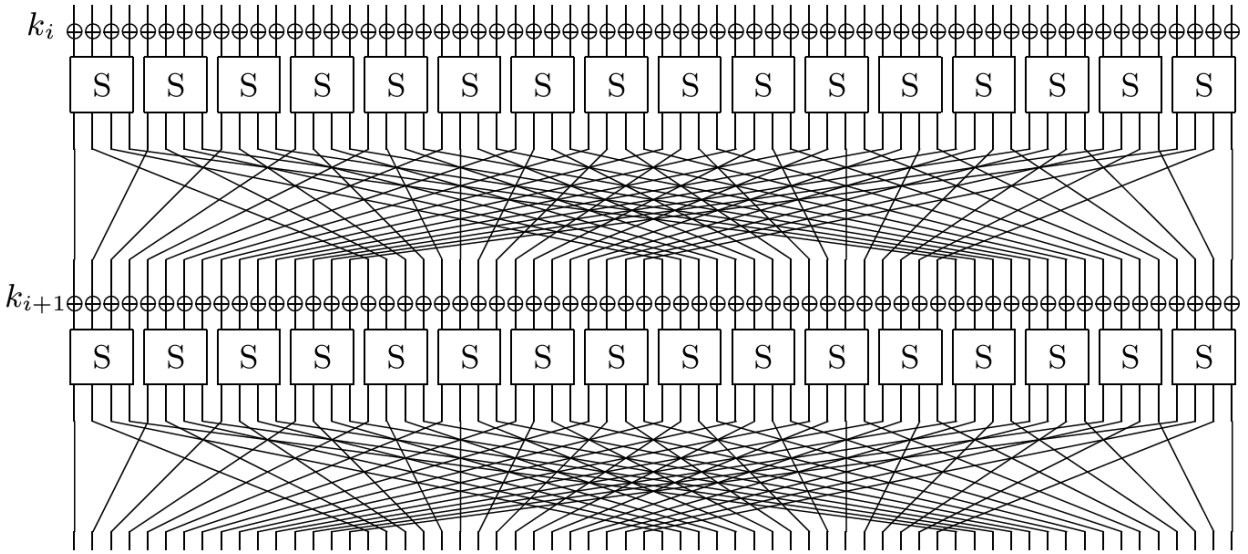
$$
\begin{aligned}
y_1 &= x_1x_2x_4 + x_1x_3x_4 \\
&\quad + x_1 + x_2x_3x_4 + x_2x_3 + x_3 + x_4 + 1 \\
y_2 &= x_1x_2x_4 + x_1x_3x_4 + x_1x_3 + x_1x_4 + \\
&\quad x_1 + x_2 + x_3x_4 + 1 \\
y_3 &= x_1x_2x_4 + x_1x_2 + x_1x_3x_4 + x_1x_3 + \\
&\quad x_1 + x_2x_3x_4 + x_3 \\
y_4 &= x_1 + x_2x_3 + x_2 + x_4
\end{aligned}
$$

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

PRESENT: A Lightweight Block Cipher

- PLayer

- It is basically a bit-permutation
  - But it has also got certain properties
  - Especially, it should ensure high number of **active S-Boxes**
  - **No cost in hardware…But tricky to implement in software**

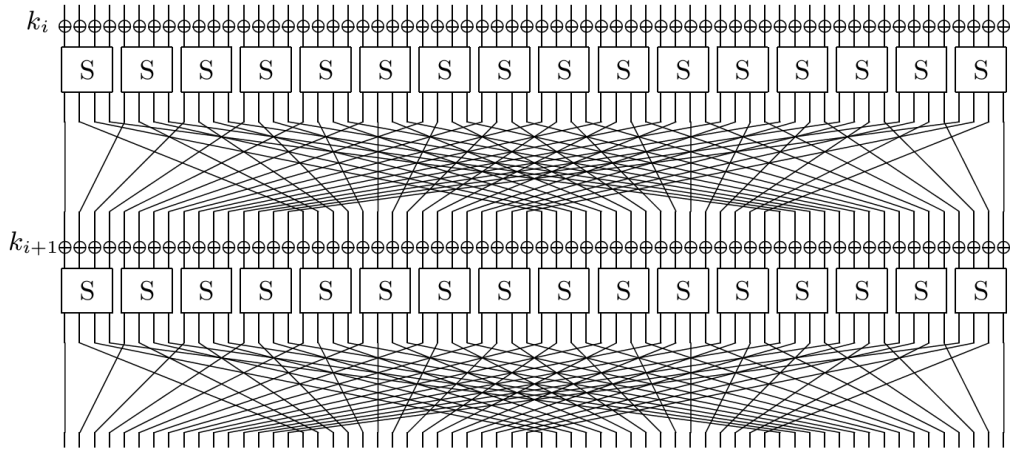


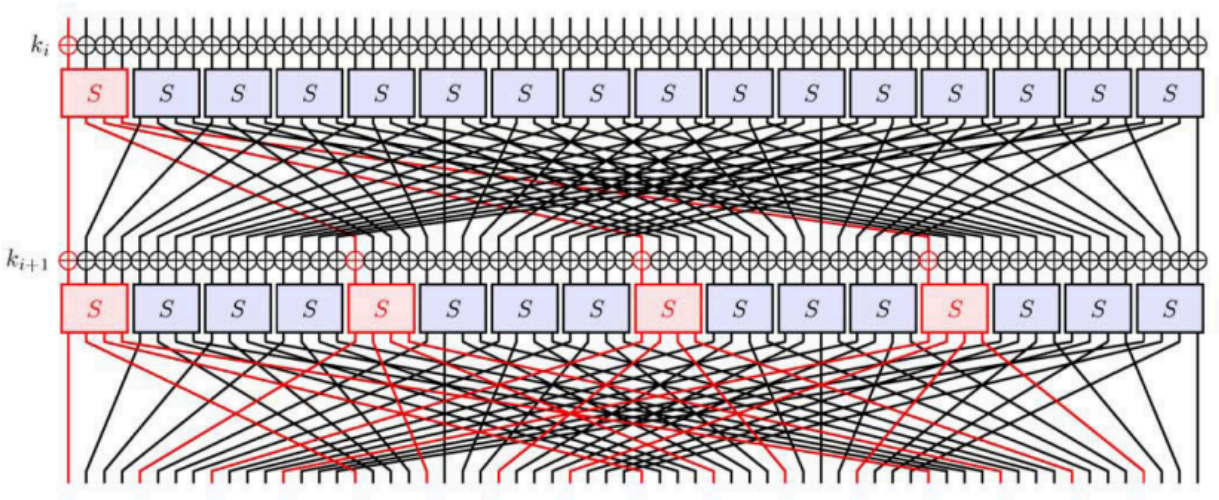
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

PRESENT: A Lightweight Block Cipher

- Key Schedule

- It is basically a bit-permutation
  - But it has also got certain properties
  - Especially, it should ensure high number of **active S-Boxes**
  - **No cost in hardware…But tricky to implement in software**





i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

PRESENT: A Lightweight Block Cipher

- Key Schedule

- Let us denote the 80-bit master key as $K = k_{79}k_{78}\cdots k_0$
- Round key: $K_i = \kappa_{63}\kappa_{62}\cdots\kappa_0 = k_{79}k_{78}\cdots k_{16}$
- Key update:

1. $[\mathtt{k}_{79}\mathtt{k}_{78}\ldots\mathtt{k}_1\mathtt{k}_0] = [\mathtt{k}_{18}\mathtt{k}_{17}\ldots\mathtt{k}_{20}\mathtt{k}_{19}]$
2. $[\mathtt{k}_{79}\mathtt{k}_{78}\mathtt{k}_{77}\mathtt{k}_{76}] = S[\mathtt{k}_{79}\mathtt{k}_{78}\mathtt{k}_{77}\mathtt{k}_{76}]$
3. $[\mathtt{k}_{19}\mathtt{k}_{18}\mathtt{k}_{17}\mathtt{k}_{16}\mathtt{k}_{15}] = [\mathtt{k}_{19}\mathtt{k}_{18}\mathtt{k}_{17}\mathtt{k}_{16}\mathtt{k}_{15}] \oplus \mathtt{round\_counter}$

PRESENT: A Lightweight Block Cipher

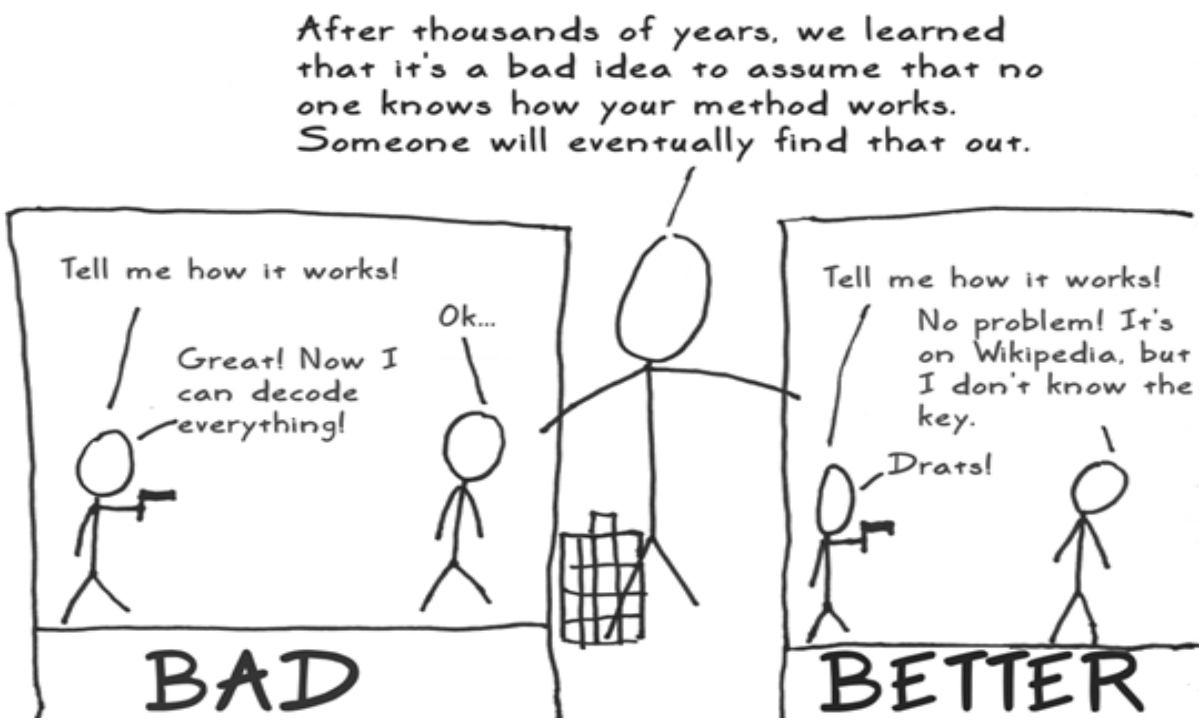
- How to Decrypt

- Just perform the sequence of operations in reverse.
- Need the inverse of the S-Box
- Need the inverse permutation
- Can you calculate them? — try it in Python or Sage

Cryptographic Attacks

- Cryptanalytic Attacks:
  - Applies mathematical techniques to obtain the key better than a brute force search (try all possibilities)
- All attacks are distinguishers:
  - all (good) ciphers transforms the plaintext distribution to “appear” as random.
  - the goal of an attack is to find properties in the cipher which does not exist in random distribution.
  - Attacker guesses a portion of the key and checks for the property.
  - Any attack better than a brute force search qualifies as an attack.
  - May not be practical but exposes design flaw.

Cryptanalysis and the Kerckhoff's Principle



- The cryptosystem is known to the adversary.
- But the key is not known to the attacker.
- The secrecy of the cryptosystem lies in the key.

- Cryptanalysis is the art of obtaining the key.

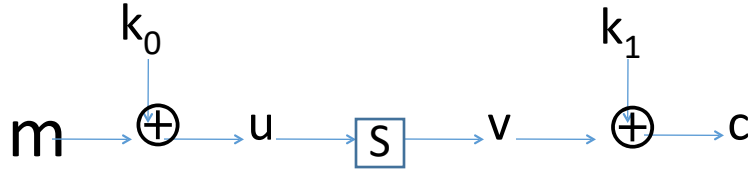
A Basic Cipher and the notion of Differential

- $c=m \oplus k$, where each variable is of b bits.
- If the key is chosen at random and used only once, then cryptanalyst gains no information about m from c.
- What happens if the same key is used twice.

  $c_0 \oplus c_1=(m_0 \oplus k) \oplus(m_1 \oplus k)=m_0 \oplus m_1$.
- Provides a notion of ‘difference’ which does not depend on the key!

An Exercise on Cryptanalysis

x	0 1 2 3 4 5 6 7 8 9 a b c d e f
S[x]	6 4 c 5 0 7 2 e 1 f 3 d 8 a 9 b

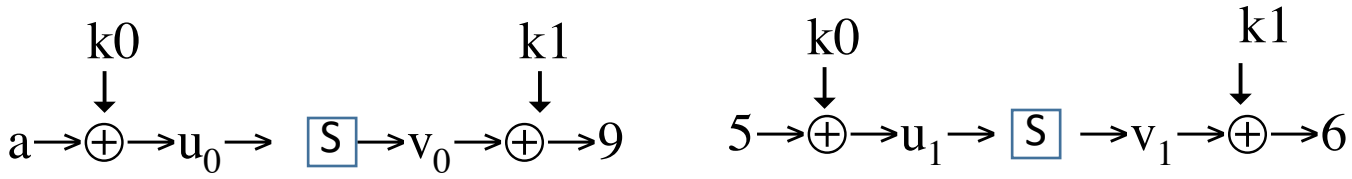


- Study the ‘differential’ behavior.
- Consider two encryptions with a pair of plaintexts, m_0 and m_1.
- The attacker knows the difference of the internal values, $u_0 \oplus u_1$.

Attack on the Cipher

- $u_0 \oplus u_1 = S^{-1}[v_0] \oplus S^{-1}[v_1]$
- Attacker guesses k_1, and estimates v_0 and v_1, and evaluates the above.
- If more than one values of k_1 satisfy, try with other messages.
- Observations:
  - Differences between internal variables are exposed.
  - Recover key information by guessing parts of the key and testing whether a differential holds.

The Differential Attack



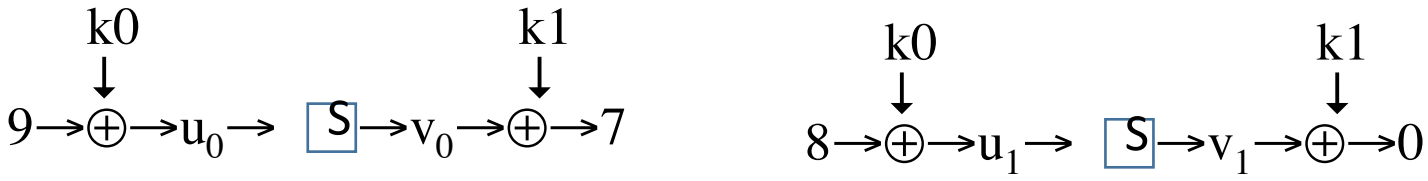
- $u' = u_0 \oplus u_1 = a \oplus 5 = f$.
- Consider the inverse R[.] from S[.]:

x	0 1 2 3 4 5 6 7 8 9 a b c d e f
R[x]	4 8 6 a 1 3 0 5 c e d f 2 b 7 9

- For each value t of k_1, check: $R[t \oplus 9] \oplus R[t \oplus 6] = f \Rightarrow k_1 \in \{7,8\}$

t	0 1 2 3 4 5 6 **7** **8** 9 a b c d e f
u'	e b e e d 8 d **f** **f** d 8 d e e b e

The Differential Attack (Contd.)



- $u_0 \oplus u_1 = 9 \oplus 8 = 1$.
- For each value t of k_1, check:

  $R[t \oplus 7] \oplus R[t \oplus 0] = 1 \Rightarrow k_1 \in \{0,7\}$

t	**0** 1 2 3 4 5 6 **7** 8 9 a b c d e f
u'	**1** 8 5 b b 5 8 **1** 5 9 2 b b 2 9 5

Thus, $k_1=7$, and $k_0=d$.