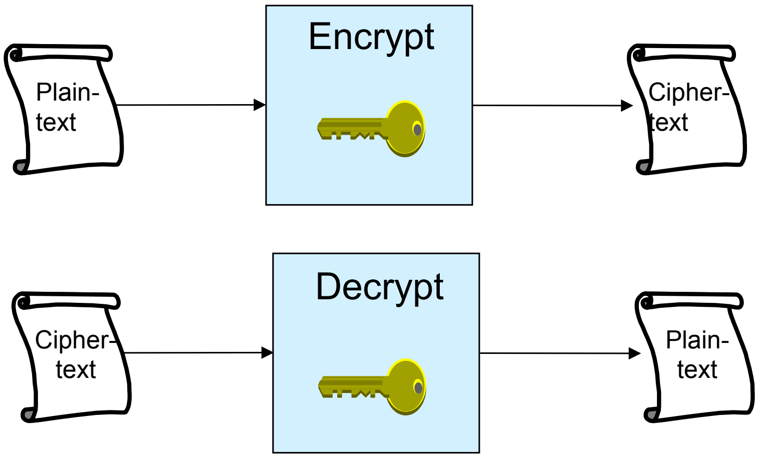
Network Security

Chapter 3
Symmetric Cryptography

- Modes of Encryption
- Data Encryption Standard (DES)
- Advanced Encryption Standard (AES)
- The Block Cipher RC4
- KASUMI

Symmetric Encryption

- General description:
  - The same key $K_{A,B}$ is used for enciphering and deciphering of messages:



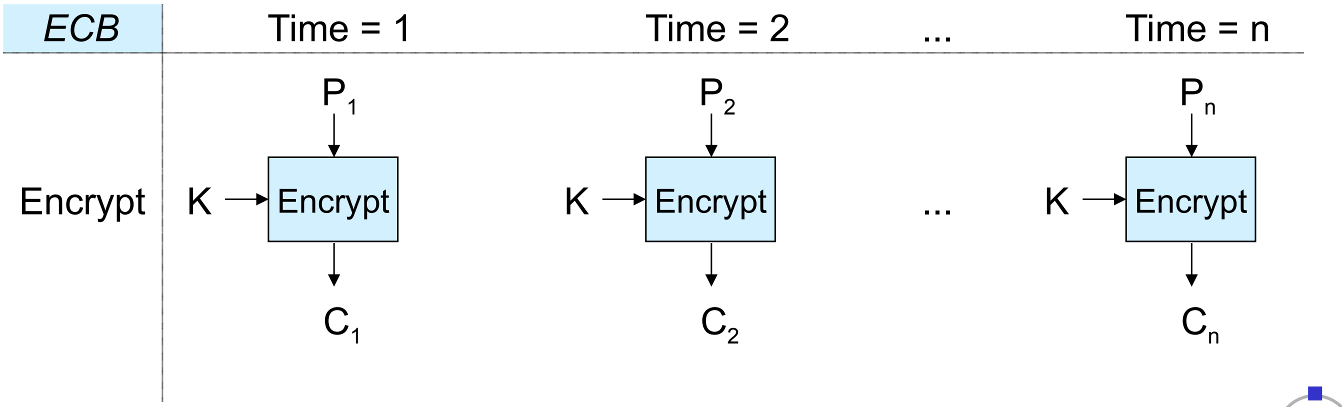
- Notation:
  - If P denotes the plaintext message $E(K_{A,B}, P)$ denotes the ciphertext and it holds $D(K_{A,B}, E(K_{A,B}, P)) = P$
  - Alternatively we sometimes write $\{P\}_{K_{A,B}}$ or $E_{K_{A,B}}(P)$ for $E(K_{A,B}, P)$
- Examples: DES, 3DES, AES, ...

Symmetric Block Ciphers - Modes of Encryption 1

- General Remarks & Notation:
  - A plaintext *P* is segmented in blocks $P_1, P_2, \dots$ each of length *b* or *j*, respectively, where *b* denotes the block size of the encryption algorithm and $j < b$
  - The ciphertext *C* is the combination of $C_1, C_2, \dots$ where c_i denotes the result of the encryption of the *i*th block of the plaintext message
  - The entities encrypting and decrypting a message have agreed upon a key *K.*

Symmetric Block Ciphers - *ECB*

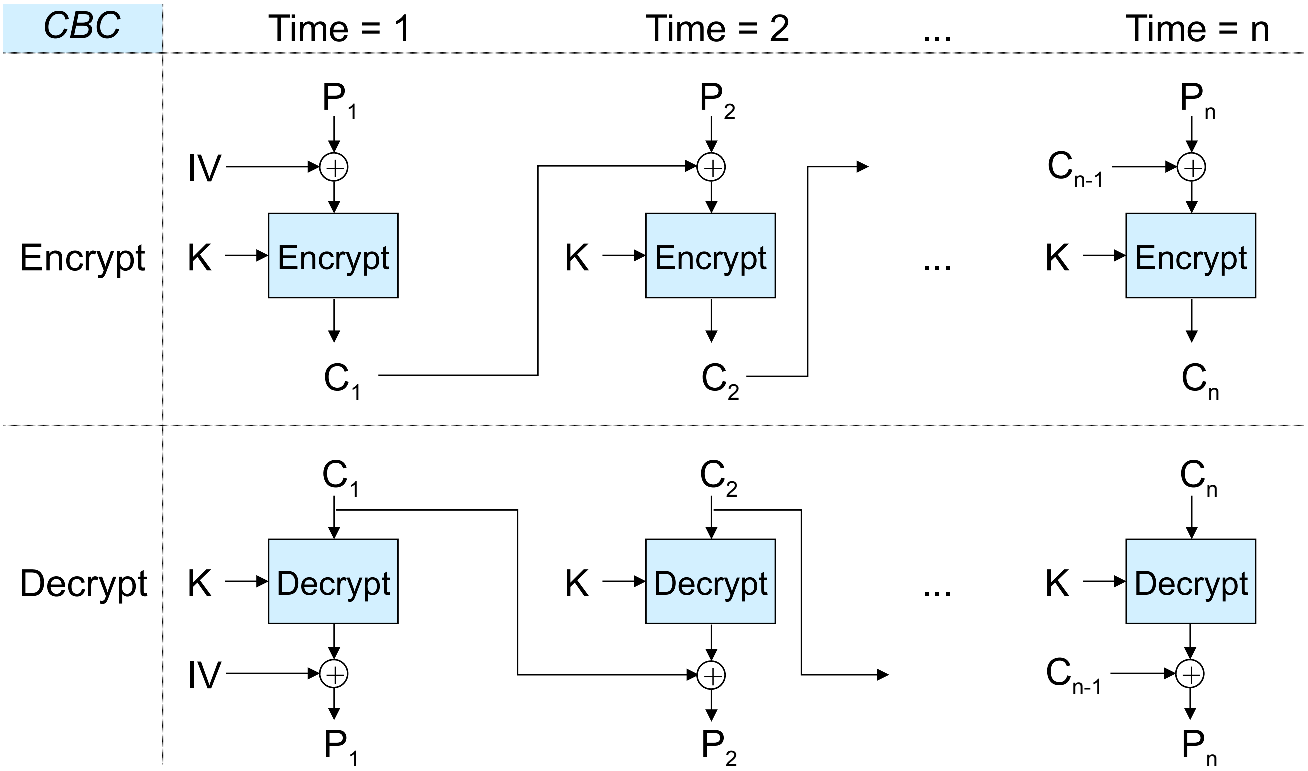
- *Electronic Code Book Mode (ECB):*
  - Every block P_i of length *b* is encrypted independently: $C_i = E(K, p_i)$
  - A bit error in one ciphertext block C_i results in a completely wrongly recovered plaintext block $P_i´$
  - Loss of synchronization does not have any effect if integer multiples of the block size *b* are lost.
    If any other number of bits are lost, explicit re-synchronization is needed.
  - Drawback: identical plaintext blocks are encrypted to identical ciphertext!



Symmetric Block Ciphers - CBC

- *Cipher Block Chaining Mode (CBC):*
  - Before encrypting a plaintext block p_i it is XORed ($\oplus$) with the preceding ciphertext block C_{i-1}:
    - $C_i = E(K, C_{i-1} \oplus p_i)$
    - $P_i' = C_{i-1} \oplus D(K, C_i)$
  - In order to compute C_1 both parties agree on an *initial value (IV)* for C_0
- Properties:
  - Error propagation:
    - A distorted ciphertext block results in two distorted plaintext blocks, as P_i' is computed using C_{i-1} and C_i
  - Synchronization:
    - If the number of lost bits is a multiple integer of *b,* one additional block P_{i+1} is distorted before synchronization is re-established.
      If any other number of bits are lost explicit re-synchronization is needed.
  - Advantage: identical plaintext blocks are encrypted to non-identical ciphertext.

Symmetric Block Ciphers - CBC 2



Symmetric Block Ciphers - CFB

- *Ciphertext Feedback Mode (CFB):*
  - A block encryption algorithm working on blocks of size *b* can be converted to an algorithm working on blocks of size *j (j<b):*
    - Let: *S(j, x)* denote the *j* higher significant bits of *x*
      P_i, C_i denote the *i^th* block of plain- and ciphertext of length *j IV* be an initial value both parties have agreed upon $R_1 = IV$
      then : $R_n = \left(R_{n-1} \cdot 2^j \bmod 2^b\right) \oplus C_{n-1}$ // j-bit left shift and XOR with old ciphertext

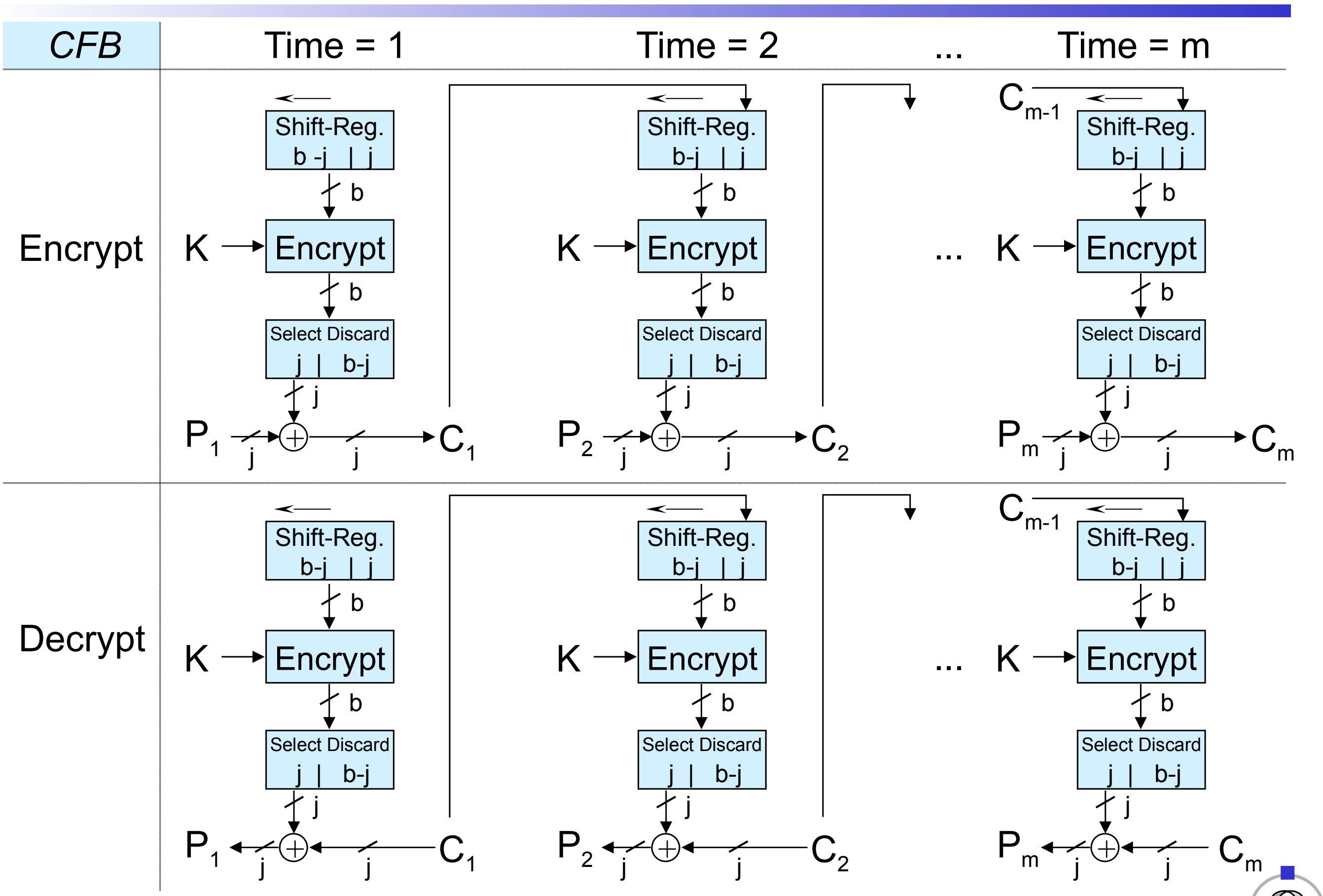
$$C_n = S\left(j, E_K\left(R_n\right)\right) \oplus P_n$$

$$S\left(j, E_K\left(R_n\right)\right) \oplus C_n = S\left(j, E_K\left(R_n\right)\right) \oplus S\left(j, E_K\left(R_n\right)\right) \oplus P_n$$

$$S\left(j, E_K\left(R_n\right)\right) \oplus C_n = P_n$$

  - A current value of *j* is 8 for encryption of one character per step

Symmetric Block Ciphers - CFB 2



Symmetric Block Ciphers - CFB 3

- Properties of CFB:
  - Error propagation:
    - As the ciphertext blocks are shifted through the register step by step, an erroneous block C_i distorts the recovered plaintext block P_i' as well as the following $\lceil b / j \rceil$ blocks
  - Synchronization:
    - If the number of lost bits is a multiple integer of *j* then $\lceil b / j \rceil$ additional blocks are distorted before synchronization is re-established.
      If any other number of bits are lost explicit re-synchronization is needed.
  - Drawback:
    - The encryption function *E* needs to be computed more often, as one encryption of *b* bit has to be performed to conceal *j* bit of plaintext
    - Example: Use of DES with encryption of one character at a time:
      ⇒ encryption has to be performed 8 times more often

Symmetric Block Ciphers - OFB

- *Output Feedback Mode (OFB):*
  - The block encryption algorithm is used to generate a pseudo-random sequence R_i, that depends only on *K* and *IV:*
    - Let: *S(j, x)* denote the *j* higher significant bits of *x*
      P_i, C_i denote the i^{th} block of plain- and ciphertext of length *j*
      IV be an initial value both parties have agreed upon
      then : $R_1 = IV$

$$R_n = \left(R_{n-1} \cdot 2^j \bmod 2^b\right) \oplus S\left(j, E_K\left(R_{n-1}\right)\right) \quad \text{// j-bit left shift + encrypted old value}$$

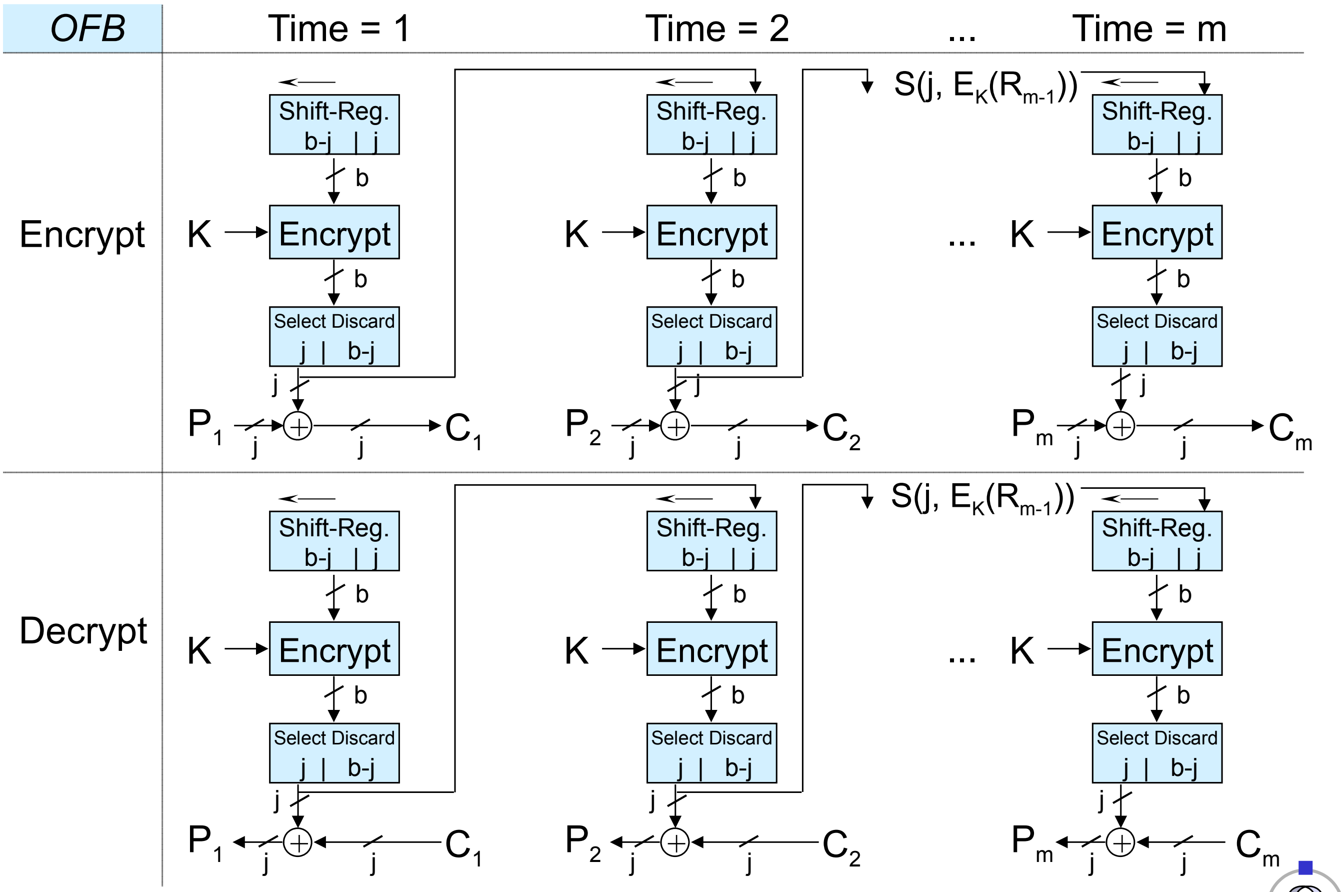
$$C_n = S\left(j, E_K\left(R_n\right)\right) \oplus P_n$$

$$S\left(j, E_K\left(R_n\right)\right) \oplus C_n = S\left(j, E_K\left(R_n\right)\right) \oplus S\left(j, E_K\left(R_n\right)\right) \oplus P_n$$

$$S\left(j, E_K\left(R_n\right)\right) \oplus C_n = P_n$$

- The plaintext is XORed with the pseudo-random sequence to obtain the ciphertext and vice versa

Symmetric Block Ciphers - OFB 2



Symmetric Block Ciphers - OFB 3

- Properties of OFB:
  - Error propagation:
    - Single bit errors result only in single bit errors ⇒ no error multiplication
  - Synchronisation:
    - If some bits are lost explicit re-synchronization is needed
  - Advantage:
    - The pseudo-random sequence can be pre-computed in order to keep the impact of encryption to the end-to-end delay low
  - Drawbacks:
    - Like with CFB the encryption function *E* needs to be computed more often, as one encryption of *b* bit has to be performed to conceal *j* bit of plaintext
    - It is possible for an attacker to manipulate specific bits of the plaintext

Symmetric Block Ciphers - Algorithm Overview

- Data Encryption Standard (DES)
  - Old American Standard from the 70s
  - Insecure because of key and block length
  - Fundamental design
  - Triple encryption with a block cipher, e.g. Triple-DES
- Advanced Encryption Standard (AES)
  - Open standardization process with international participation
  - In October 2000, one algorithm called *Rijndael* has been proposed for AES
  - AES standard announced in November 2001
  - See also http://www.nist.gov/aes/
- Other popular algorithms:
  - RC4
  - KASUMI

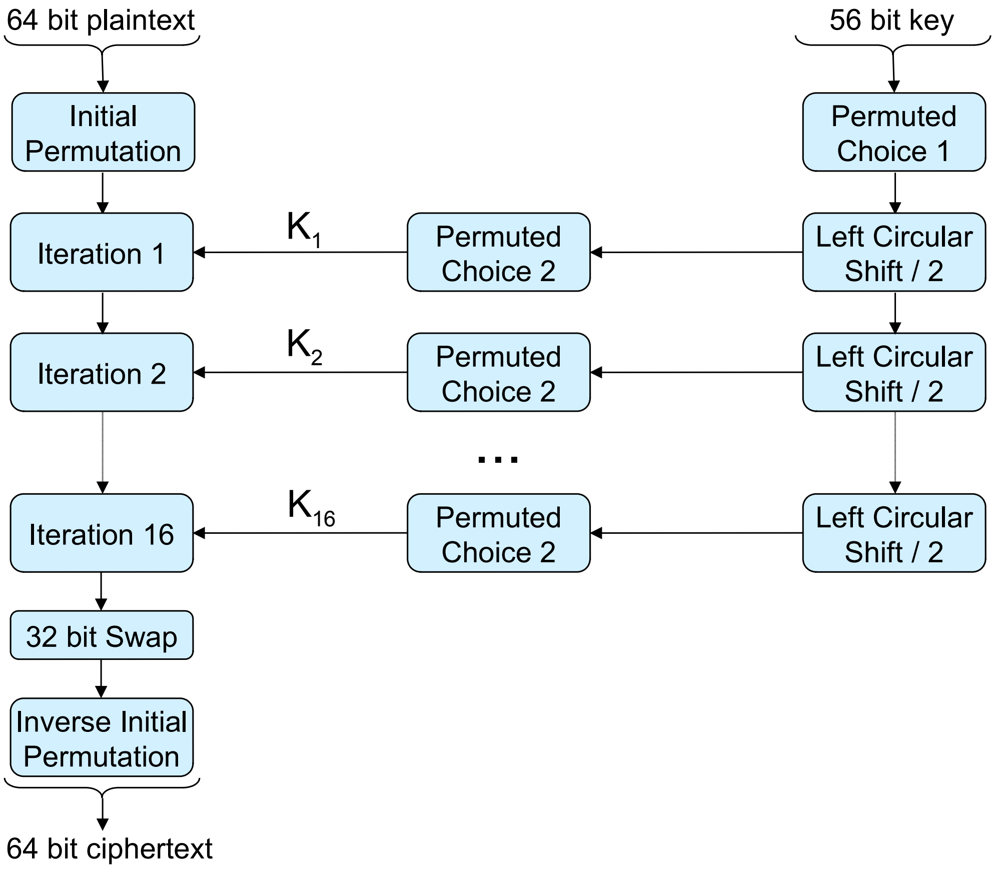
The Data Encryption Standard (DES) – History

- 1973 the National Bureau of Standards (NBS, now National Institute of Standards and Technology, NIST) issued a request for proposals for a national cipher standard, demanding the algorithm to:
  - provide a high level of security,
  - be completely specified and easy to understand,
  - provide security only by its' key and not by its' own secrecy,
  - be available to all users,
  - be adaptable for use in diverse applications,
  - be economically implementable in electronic devices,
  - be efficient to use,
  - be able to be validated, and
  - be exportable.
- None of the submissions to this first call came close to these criteria.
- In response to a second call, IBM submitted its' algorithm LUCIFER, a symmetric block cipher, which works on blocks of length 128 bit using keys of length 128 bit and that was the only promising candidate

DES – History continued

- The NBS requested the help of the National Security Agency (NSA) in evaluating the algorithm’s security:
  - The NSA reduced the block size to 64 bit, the size of the key to 56 bit and changed details in the algorithm’s *substitution boxes.*
  - Many of the NSA’s reasoning for these modifications became clear in the early 1990’s, but raised great concern in the late 1970’s.
- Despite all criticism the algorithm was adopted as “Data Encryption Standard” in the series of Federal Information Processing Standards in 1977 (FIPS PUB 46) and authorized for use on all unclassified government communications.
- DES has been widely adopted in the years to follow

DES – Algorithm Outline



64 bit plaintext

Initial Permutation

Iteration 1

Iteration 2

Iteration 16

32 bit Swap

Inverse Initial Permutation

64 bit ciphertext

K_1

K_2

…

K_{16}

Permuted Choice 2

Permuted Choice 2

Permuted Choice 2

56 bit key

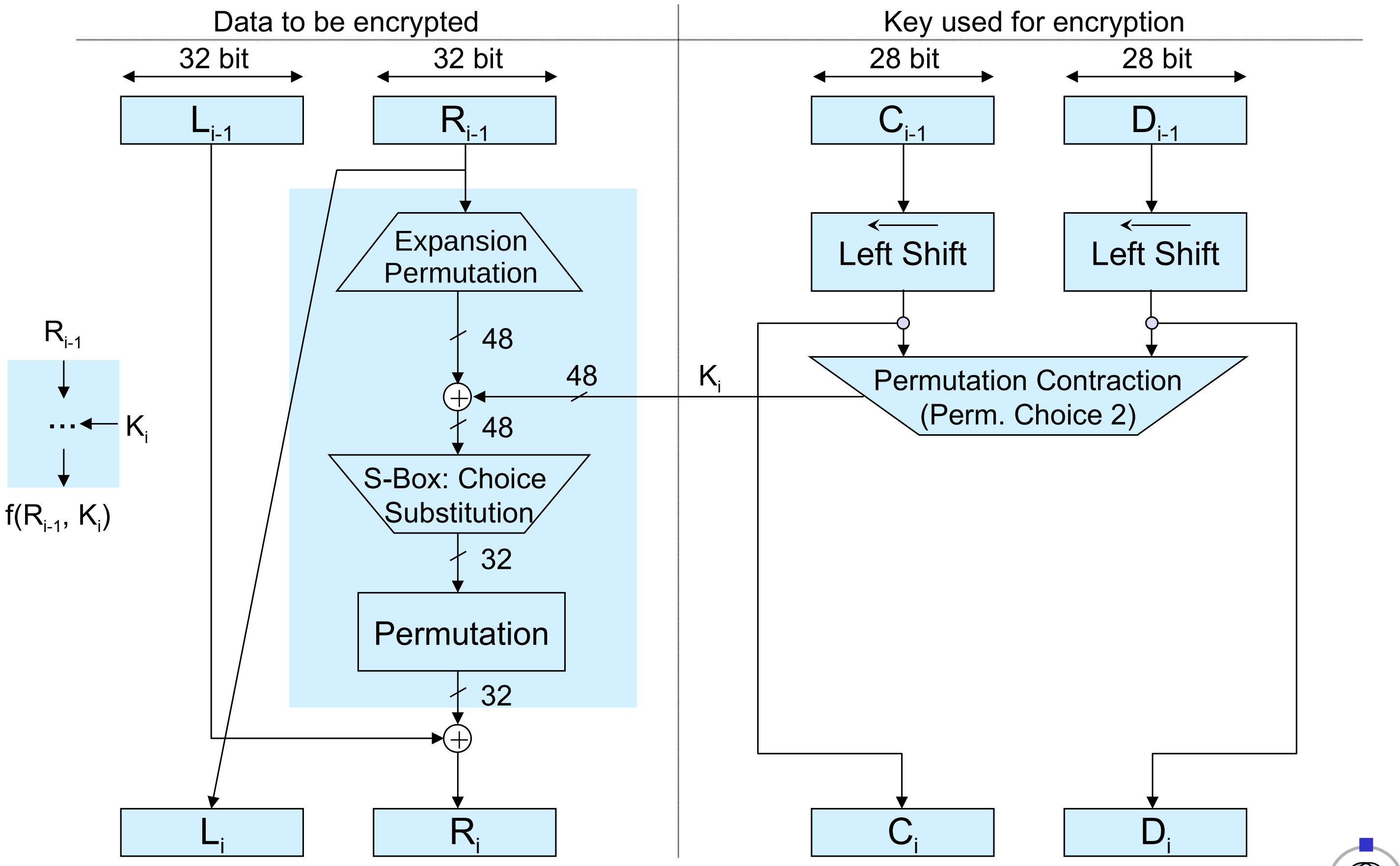
Permuted Choice 1

Left Circular Shift / 2

Left Circular Shift / 2

Left Circular Shift / 2

DES – Single Iteration (1)



DES – Single Iteration (2)

- The right-hand 32 bit of the data to be encrypted are expanded to 48 bit by the use of an expansion / permutation table
- Both the left- and the right-hand 28 bit of the key (also called *subkeys*) are circularly left-shifted and the resulting value is contracted to 48 bit by the use of a permutation / contraction table
- The above two values are XORed and fed into a choice and substitution box:
  - Internally this operation is realized by 8 so-called *s-boxes,* each of them mapping a six bit value to a four bit value according to a box-specific table, altogether leading to a 32 bit output
  - The design of these s-boxes was strengthened by the NSA, which led to intense discussion in the 1970’s and was understood in the 1990’s after the discovery of *differential cryptanalysis*
- The output of the above step is permuted again and XORed with the left-hand 32 bit of data leading to the new right-hand 32 bit of data
- The new left-hand 32 bit of data are the right-hand value of the previous iteration

DES – Decryption (1)

- Using the abbreviation *f(R, K)* the encryption process can be written as:
  - $L_i = R_{i-1}$
  - $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$
  - This design idea (splitting the data into two halfs and organize encryption according to the above equations) is used in many block ciphers and is called a *Feistel network* (after its inventor H. Feistel)
- The DES decryption process is essentially the same as encryption. It uses the ciphertext as input to the encryption algorithm, but applies the subkeys in reverse order
- So, the initial values are:
  - $L'_0 \,||\, R'_0$ = InitialPermutation(ciphertext)
  - ciphertext = InverseInitialPermutation($R_{16} \,||\, L_{16}$)
  - $L'_0 \,||\, R'_0$ = InitialPermuation(InverseInitialPermutation($R_{16} \,||\, L_{16}$)) = $R_{16} \,||\, L_{16}$
- After one step of decryption:
  - $L'_1 = R'_0 = L_{16} = R_{15}$
  - $R'_1 = L'_0 \oplus f(R'_0, K_{16}) = R_{16} \oplus f(R_{15}, K_{16}) = [L_{15} \oplus f(R_{15}, K_{16})] \oplus f(R_{15}, K_{16}) = L_{15}$

DES – Decryption (2)

- This relationship holds through all the process as:
  - $R_{i-1} = L_i$
  - $L_{i-1} = R_i \oplus f(R_{i-1}, K_i) = R_i \oplus f(L_i, K_i)$
- Finally, the output of the last round is:
  - $L'_{16} \,||\, R'_{16} = R_0 \,||\, L_0$
- After the last round, DES performs a 32-bit swap and the inverse initial permutation:
  - *InverseInitialPermutation($L_0 \,||\, R_0$) = InverseInitialPermutation(InitialPermutation(plaintext)) = plaintext*

DES – Security (1)

- Key weaknesses:
  - *Weak keys:* four keys are weak as they generate subkeys with either all 0's or all 1's
  - *Semiweak keys:* there are six pairs of keys, which encrypt plaintext to identical ciphertext as they generate only two different subkeys
  - *Possibly weak keys:* there are 48 keys, which generate only four different subkeys
  - As a whole 64 keys out of 72,057,594,037,927,936 are considered weak
- Algebraic structure:
  - If DES were *closed,* then for every K_1, K_2 there would be a K_3 such that: $E(K_2, E(K_1,M)) = E(K_3, M)$, thus double encryption would be useless
  - If DES were *pure,* Then for every K_1, K_2, K_3 there would be a K_4 such that $E(K_3, E(K_2, E(K_1, M))) = E(K_4, M)$ thus triple encryption would be useless
  - DES is neither *closed* nor *pure,* thus a multiple encryption scheme might be used to increase the key length (see also below)

DES – Security (2)

- *Differential cryptanalysis:*
  - In 1990 E. Biham and A. Shamir published this method of analysis
  - It looks specifically for differences in ciphertexts whose plaintexts have particular differences and tries to guess the correct key from this
  - The basic approach needs chosen plaintext together with its ciphertext
  - DES with 16 rounds is immune against this attack, as the attack needs 2^{47} chosen plaintexts or (when "converted" to a known plaintext attack) 2^{55} known plaintexts.
  - The designers of DES told in the 1990's that they knew about this kind of attacks in the 1970's and that the s-boxes were designed accordingly
- Key length:
  - As a 56 bit key can be searched in 10.01 hours when being able to perform 10^6 encryptions / μs (which is feasible today), DES can no longer be considered as sufficiently secure

Extending the Key-Length of DES by Multiple Encryption (1)

- Double DES: as DES is not closed, double encryption results in a cipher that uses 112 bit keys:
  - Unfortunately, it can be attacked with an effort of 2^{56}
  - As $C = E(K_2, E(K_1, P))$ we have $X := E(K_1, P) = D(K_2, C)$
  - If an attacker can get one known plaintext / ciphertext pair then he can construct two tables (*meet-in-the-middle-attack*):
    - Table 1 holds the values of *X* when *P* is encrypted with all possible values of *K*
    - Table 2 holds the values of *X* when *C* is decrypted with all possible values of *K*
    - Sort the two tables and construct keys $K_{T1} || K_{T2}$ for all combinations of entries that yield to the same value
  - As there are 2^{64} possible ciphertext values for any given plaintext that could be produced by Double-DES, there will be on the average $2^{112}/2^{64} = 2^{48}$ false alarms on the first known plaintext / ciphertext pair.
  - Every additional plaintext / ciphertext pair reduces the chance of getting a wrong key by a factor of $1 / 2^{64}$, so with two known blocks the chance is 2^{-16}

Extending the Key-Length of DES by Multiple Encryption (2)

- So, the effort required to break Double DES is on the magnitude of 2^{56}, which is only slightly better than the effort of 2^{55} required to break Single DES with a known plaintext attack and far from the 2^{111} we would expect from cipher with a key length of 112 bit!
- This kind of attack can be circumvented by using a triple encryption scheme, as proposed by W. Tuchman in 1979:
  - $C = E(K_3, D(K_2, E(K_1, P)))$
  - The use of the decryption function *D* in the middle allows to use triple encryption devices with peers that only own single encryption devices by setting $K_1 = K_2 = K_3$
  - Triple encryption can be used with two (set $K_1 = K_3$) or three different keys
  - There are no known practical attacks against this scheme up to now
  - Drawback: the performance is only 1/3 of that of single encryption, so it might be a better idea to use a different cipher, which offers a bigger key-length right away

The Advanced Encryption Standard AES (1)

- Jan. 1997: the *National Institute of Standards and Technology (NIST)* of the USA announces *the AES development* effort.
  - The overall goal is to develop a Federal Information Processing Standard (FIPS) that specifies an encryption algorithm(s) capable of protecting sensitive government information well into the next century.
- Sep. 1997: formal *call for algorithms*, open to everyone on earth
  - AES would specify an unclassified, publicly disclosed encryption algorithm(s), available royalty-free, worldwide.
- Aug. 1998: first AES candidate conference
  - NIST announces the selection of 15 candidate algorithms
  - Demand for public comments
- April 1999:
  - Using the analyses and comments received, NIST selects five algorithms as finalist candidates: *MARS, RC6, Rijndael, Serpent,* and *Twofish*
- October 2000: Rijndael is announced as NIST’s proposal for AES
- 26. November 2001: official announcement of the AES standard

The Advanced Encryption Standard AES (2)

- Round-based symmetric cipher
- No Feistel structure (different encryption and decryption functions)
- Key and block lengths:
  - Key length: 128, 192, or 256 bit
  - Block length: 128, 192, or 256 bit (only 128 bit version standardized)
  - Number of rounds: 10, 12, 14

Standardized AES Configurations

Key Size [bit]	Block Length [bit]	# Rounds
128	128	10
192	128	12
256	128	14

The Advanced Encryption Standard AES (3)

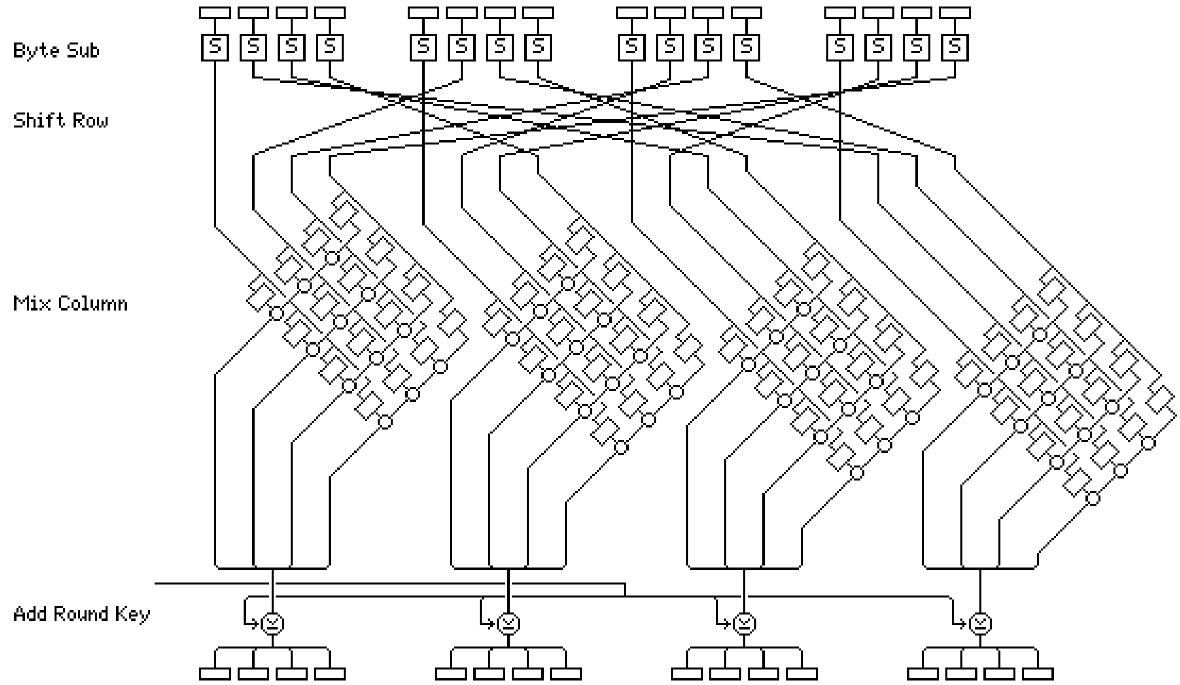
- The algorithm operates on:
  - state[4, 4]: a byte-array of 4 rows and 4 columns (for 128 bit block size)
  - key[4, 4]: an array of 4 rows and 4 columns (for 128 bit key size)
- Encryption: (for block and key size of 128 bit)
  - Rounds 1 - 9 make use of four different operations:
    - ByteSub: a non-linear byte substitution by a fixed table (basically an s-box)
    - ShiftRow: the rows of the state are cyclically shifted by various offsets
    - MixColumn: the columns of state[] are considered as polynomials over GF(2^8) and multiplied modulo $x^4 + 1$ with a fixed matrix:

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}$$

    - RoundKey: a round-key is XORed with the state
  - Round 10 does not make use of the MixColumn operation

The Advanced Encryption Standard AES (4)

Structure of one Round in Rijndael



(source: “Rijndael”, a presentation by J. Daemen and V. Rijmen)

The Advanced Encryption Standard AES (5)

- Decryption:
  - Round-keys and operations applied in reverse order
  - MixColumn step can only be inversed by multiplying with the inverse matrix (also over GF(2^8))

$$\begin{pmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{pmatrix}$$

  - Often tabularized pre-calculated solutions are used, but takes more space

AES – Security

- The simple mathematical structure of AES is the major reason for its speed, but led to criticism
- Only the ByteSub function is really non-linear and prevents effective analysis
- AES may be described as a large matrix operation
- Already during standardization attacks for reduced versions have been developed
  - An attack with 2^{32} chosen plaintext against a 7 round version of AES [GM00]
  - Significant reduction of complexity even for a 9 round version of AES with 256 key size with a related key attack
- 2011 the first attack against a full AES became known [BKR11]
  - Key recovery in $2^{126.1}$ for AES with 128 bits, $2^{189.7}$ for AES with 192 bits, $2^{254.4}$ for AES with 256 bits
  - "Practical" attack (does not assume related keys), **but**
  - only a small scratch when considering 10 years of cryptographic research

The Stream Cipher Algorithm RC4 (1)

- RC4 is a stream cipher that has been invented by Ron Rivest in 1987
- It was proprietary until 1994 when someone posted it anonymously to a mailing list
- RC4 is operated in the output feedback mode (OFB):
  - The encryption algorithm generates a pseudo-random sequence RC4(IV, K), that depends only on the key K and an initialization vector IV
  - The plaintext P_i is then XORed with the pseudo-random sequence to obtain the ciphertext and vice versa:
    - $C_1 = P_1 \oplus$ RC4(IV_1, K)
    - $P_1 = C_1 \oplus$ RC4(IV_1, K)
  - The pseudo-random sequence is often also called *keystream*
  - It is crucial to the security that keystream is never re-used!!!
    - If keystream is re-used (that is $IV_1 = IV_2$ with the same K), then the XOR of two plaintexts can be obtained:
      $C_1 \oplus C_2 = P_1 \oplus$ RC4$(IV, K) \oplus P_2 \oplus$ RC4$(IV, K) = P_1 \oplus P_2$

The Stream Cipher Algorithm RC4 (2)

- RC4 uses a variable length key up to 2048 bit
  - Actually, the key serves as the seed for a pseudo-random-bit-generator
- RC4 works with two 256 byte arrays: S[0,255], K[0,255]
- Step 1: Initialize the arrays

```
for (i = 0; i < 256; i++) S[i] = i;   // fill array S[] with 0 to 255
 // fill array K[] with the key and IV by repeating them until K[] is filled
n = 0;
for (i =0; i < 256; i++) { n = (n + S[i] + K[i]) MOD 256; swap(S[i], S[n]); }
```

- Step 2: Generate the keystream (after initializing i = 0; n = 0;)

```
i = (i + 1) MOD 256; n = (n + S[i])  MOD 256;
swap(S[i], S[n]);
t = (S[i] + S[n]) MOD 256;
Z = S[t]; // Z contains 8 bit of keystream produced by one iteration
```

- Step 3: XOR the keystream with the plaintext or ciphertext

The Stream Cipher Algorithm RC4 (3)

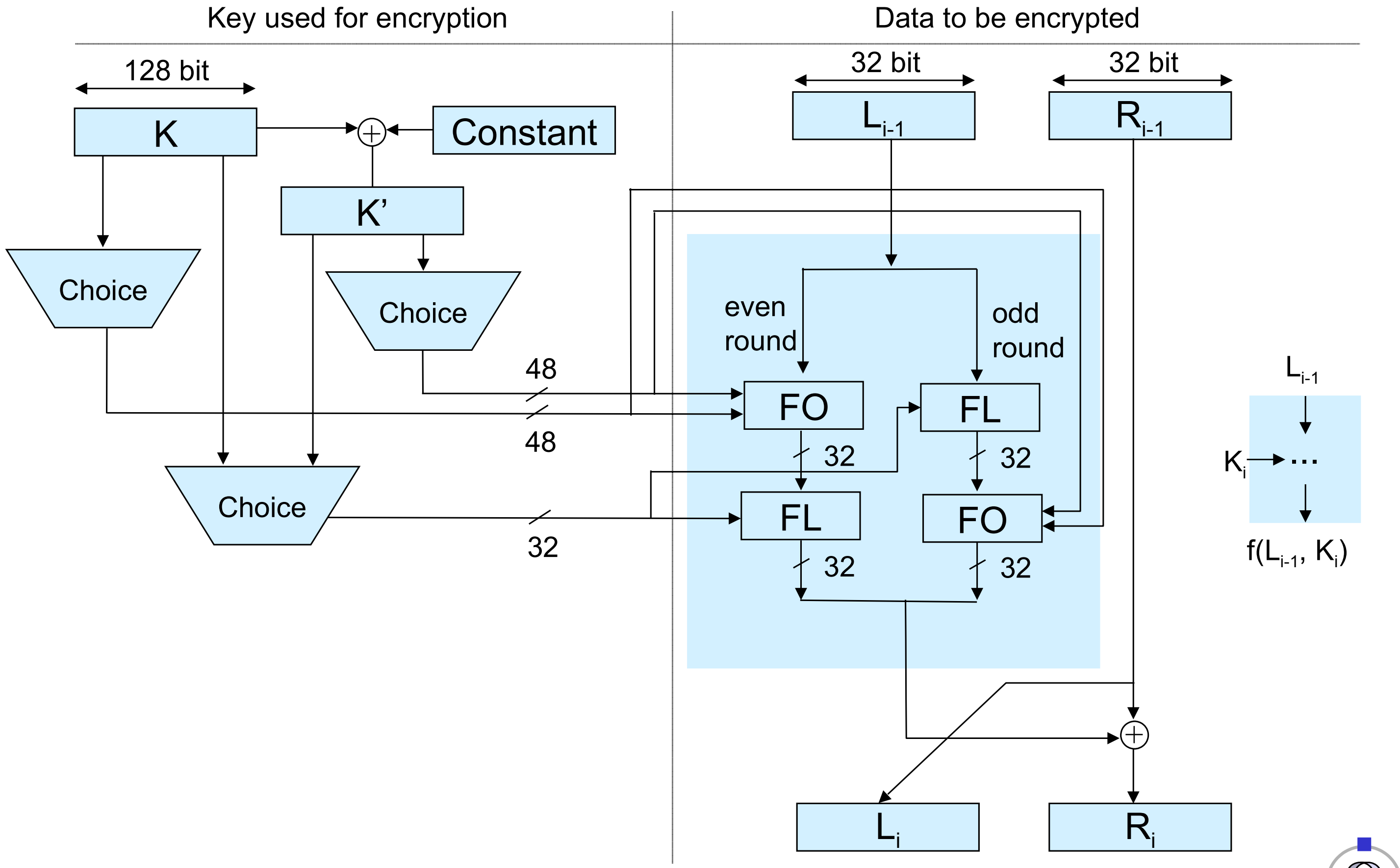
- Security of RC4:
  - Security against brute force attacks (trying every possible key):
    - The variable key length of up to 2048 bit allows to make them impractical (at least with the resources available in our universe)
    - However, by reducing the key length RC4 can also be made arbitrarily insecure!
  - RSA Data Security, Inc. claims that RC4 is immune to differential and linear cryptanalysis, and no small cycles are known
- RC4 with 40 bit keys had special export status, even when other ciphers were not allowed to be exported from the USA
  - Secure Socket Layer (SSL) used RC4 with 40 bit keys as default algorithm
  - 40 bit key length is not immune against brute-force attacks
- However, depending on the key scheduling method, RC4 may be severely vulnerable! [FMS01a, Riv01a, SIR01a]
- It is recommended to discard at least the first 3072 bytes of the key stream [Mir02, Kle08]
- Should not really be used any more, even with longer keys

KASUMI

- Used to encrypt calls in GSM and UMTS, implements f(8) and f(9) (also called A5/3, UEA1, UIA1)
- Initially standardized by 3GPP in 2000 [ETS12] and based on MISTY1 by Mitsubishi
- Designed for hardware implementation
  - Fast implementation possible
  - < 10k gates

- 64 bit block size
- 128 bit key length
- 8 round Feistel network
- Safety margin not very large

KASUMI – Single Iteration (1)



KASUMI – Single Iteration (2)

- The left-hand 32 bit of the data to be encrypted is modified by two non-linear functions FO and FL that both use keying material
- The order in which FO and FL are applied depends on the round number
- FL splits data into 16 bit words that are combined with keying material, permutated, and XORed with the original values
- FO is a 3-round Feistel network with a modifying function FI that is itself a Feistel-like network that employs two s-boxes
- The output of the above step is XORed with the right-hand 32 bit of data leading to the new right-hand 32 bit of data
- The new left-hand 32 bit of data is the right-hand value of the previous iteration

KASUMI – Security Discussion

- A reduced version of KASUMI (6 rounds) can be attacked by so-called *impossible differential crypto-analysis*, where impossible states of the cipher are deducted from ciphertext/plaintext pairs
  - First published already one year after standardization
  - Time complexity of 2^{100} [Kue01]
- For a full version of KASUMI related key attacks are possible
  - Chosen plain-text attack, where attacker can encrypt the same data with multiple "related" keys
  - Time complexity of $2^{76.1}$ [BDN05] and 2^{32} at best [DKS10]
  - However, conditions where attackers have access to related keys in 3G networks are very seldom
  - Interestingly MISTY is not affected by these attacks!
- However ETSI has adopted SNOW 3G (UEA2 and UIA2) [ETS06] to be prepared for a full breach of KASUMI
  - Stream cipher based on LFSR, can be implemented in 7,500 ASIC gates
  - But also vulnerable to related key attacks [KY11]

Additional References

[AES01a] National Institute of Standards and Technology (NIST). *Specification for the Advanced Encryption Standard (AES).* Federal Information Processing Standards Publication, February 2001.

[DR97a] J. Daemen, V. Rijmen. *AES Proposal: Rijndael.* http://csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf, 1997.

[FMS01a] S. Fluhrer, I. Mantin, A. Shamir. *Weaknesses in the Key Scheduling Algorithm of RC4.* Eighth Annual Workshop on Selected Areas in Cryptography, August 2001.

[Riv01a] R. Rivest. *RSA Security Response to Weaknesses in Key Scheduling Algorithm of RC4*. http://www. rsa.com/rsalabs/node.asp?id=2009, 2001.

[SIR01a] A. Stubblefield, J. Ioannidis, A. D. Rubin. *Using the Fluhrer, Mantin, and Shamir Attack to Break WEP.* AT&T Labs Technical Report TD-4ZCPZZ, August 2001.

[FKLS00] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D.Wagner, D. Whiting. *Improved cryptanalysis of Rijndael*. In FSE'00, volume 1978 of Lecture Notes in Computer Science, pages 213–230. Springer, 2000.

Additional References

[GM00] H. Gilbert and M. Minier. A Collision Attack on 7 Rounds of Rijndael. In AES Candidate Conference, pages 230–241, 2000.

[BKR11] A. Bogdanov, D. Khovratovich, C. Rechberger. *Biclique cryptanalysis of the full AES*. In ASIACRYPT'11, pages 344-371, 2001.

[Mir02] I. Mironov. *(Not so) random shuffles of RC4.* In Advances in Cryptology – CRYPTO 2002, volume 2442 of LNCS, pages 304–319, 2002.

[Kle08] A.Klein. *Attacks on the RC4 stream cipher*. In Designs, Codes and Cryptography. 48, volume 3, pages 269-286, 2008.

[ETS12] ETSI/SAGE. *Specification of the 3GPP confidentiality and integrity algorithms; Document 2: Kasumi specification*. Release 11. 2012

[Kue01] U. Kühn. *Cryptanalysis of Reduced-Round MISTY*. In Advances in Cryptology – EUROCRYPT 2001, 2001.

[BDN05] E. Biham, O. Dunkelman, N. Keller. *A Related-Key Rectangle Attack on the Full KASUMI*, In ASIACRYPT 2005, 2005.

[DKS10] O. Dunkelman, N. Keller, A. Shamir. *A practical-time related-key attack on the KASUMI cryptosystem used in GSM and 3G telephony*. In CRYPTO'10, 2010.

[ETS06] ETSI/SAGE. *Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 & UIA2. Document 2: SNOW 3G Specification*. Version 1.1. 2006.

[KY11] A. Kircanski A.M. Youssef. On the Sliding Property of SNOW 3G and SNOW 2.0 IET Inf. Secur., Vol. 5, Iss. 4, pages. 199–206. 2011