Handout 3

Summary of this handout: *Block Ciphers continued — AES (Rijndael) — AACS — Finite Fields — Revision: Matrix Arithmetic*

II.1.3 AES (Rijndael)

The *advanced encryption standard* (AES) is the successor of the outdated DES. It was developed by the two Belgian cryptographers Joan Daemen and Vincent Rijmen. It was originally named *Rijndael*, but renamed to AES when it was adopted as official US standard in November 2001. Rijndael is a block cipher, however, it does not rely on the basic design of the Feistel cipher. In particular is has distinct encryption and decryption algorithms. But similar to DES, it is round based and relies on a combination of substitutions, permutations and key addition. AES is motivated by arithmetic operations in the field $\mathbb{F}_{2^8}$, and its implementation in hardware and software is compact and fast.
Rijndael is parameterisable in that it can work with:

- block sizes of 128, 192, and 256 bits,
- key sizes of 128, 192, and 256 bits, and
- 10, 12, or 14 rounds of encryption.

Rijndael performs encryption, decryption and computes the key schedule using arithmetic in $\mathbb{F}_{2^8}$ with respect to the irreducible polynomial $p(x) = x^8 + x^4 + x^3 + x + 1$. That is, $\mathbb{F}_{2^8} = \mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x + 1)$.

29. Operation of AES

We will discuss the basic operations of Rijndael for the case of 128 bit block and key size and 10 rounds of encryption. Rijndael arranges both message and key in 4×4 matrices of 8-bit elements, i.e., each element is exactly one byte and each column and each row contain 32-bit words.
If $m = m_0 \| m_1 \| \dots \| m_{15}$ is the message, then Rijndael initialises the so called *state matrix* A as follows:

$$\begin{bmatrix} m_0 & m_4 & m_8 & m_{12} \\ m_1 & m_5 & m_9 & m_{13} \\ m_2 & m_6 & m_{10} & m_{14} \\ m_3 & m_7 & m_{11} & m_{15} \end{bmatrix} \longrightarrow A = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}$$

Each round applies the following manipulations to the state matrix:

1. a substitution operation on every single byte *SubBytes*,
2. a byte permutation *ShiftRows*,
3. a column manipulation *MixColumns*,
4. and an xor of the state with the round key *AddRoundKey*.

An exception is the last round, where the *MixColumns* operation is skipped.
For each of these operations, except for *AddRoundKey*, we need corresponding inverse operations for the decryption. Here is an overview of the encryption and decryption algorithms:

$E_K(M)$	$D_K(C)$
$A := M$	$A := C$
$A :=$ AddRoundKey(A, K_0)	$A :=$ AddRoundKey(A, K_{10})
for i = 1 **to** 9 **do**	$A :=$ InverseShiftRows(A)
$A :=$ SubBytes(A)	$A :=$ InverseSubBytes(A)
$A :=$ ShiftRows(A)	**for** i = 9 **downto** 1 **do**
$A :=$ MixColumns(A)	$A :=$ AddRoundKey(A, K_i)
$A :=$ AddRoundKey(A, K_i)	$A :=$ InverseMixColumns(A)
end	$A :=$ InverseShiftRows(A)
$A :=$ SubBytes(A)	$A :=$ InverseSubBytes(A)
$A :=$ ShiftRows(A)	**end**
$A :=$ AddRoundKey(A, K_{10})	$A :=$ AddRoundKey(A, K_0)
$C := A$	$M := A$

Here are the single operations in detail.

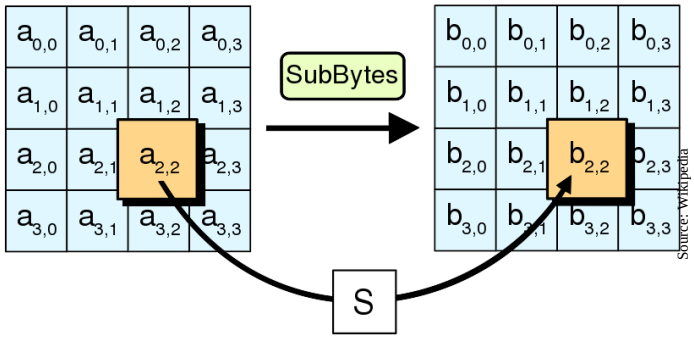
1. SubBytes: This operation is similar to the S-Box substitution of DES. Each byte $a_{i,j}$ of the state is substituted by the output of a single S-Box. This S-Box corresponds to an algebraic operation in Rijndael's finite field $\mathbb{F}_{2^8}$. Each byte $a_{i,j} = [z_7, \ldots, z_0]$ is considered as a polynomial in $\mathbb{F}_{2^8}$. Its substitution is then computed in two steps:

1. First we compute the multiplicative inverse of $a_{i,j}$ in $\mathbb{F}_{2^8}$ to get ${a_{i,j}}^{-1} = [x_7, \ldots, x_0]$. (The zero element is mapped to $[0, \ldots, 0]$.)

2. We then compute a new bit vector $b_{i,j} = [y_7, \ldots, y_0]$ with the following transformation in $\mathbb{F}_2$ (observe that the vector addition is the same as an xor $\oplus$):

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

The substitution can be schematically displayed as:

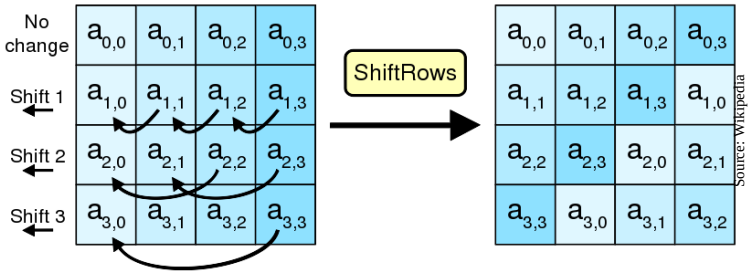


For the decryption algorithm we can define an inverse operation *InverseSubBytes* by first reversing the transformation:

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \left(\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \right)$$

and then computing the multiplicative inverse of $[x_7, \ldots, x_0]$ in $\mathbb{F}_{2^8}$.

2. ShiftRows: This operation performs a cyclic shift on the state matrix by shifting each row separately. This ensures that the columns of the state matrix interact over several rounds of encryption. The lower three rows are shifted by one, two, and three positions, respectively:



For the decryption the *InverseShiftRows* does the reverse shift.

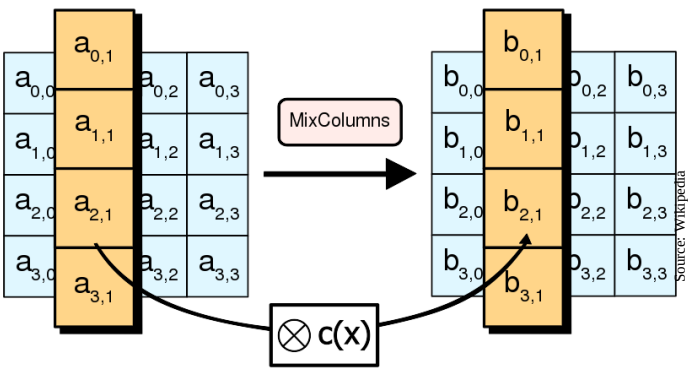
3. MixColumns: This operation ensure interaction of the rows of the state matrix by mixing each column separately. For this it performs the following matrix multiplication for each column $i = 1, 2, 3, 4$ over $\mathbb{F}_{2^8}$, where the entries of the matrix are hexadecimal representations of polynomials of degree 7 (e.g. $0x03$ corresponds to $x + 1$.)

$$\begin{bmatrix} b_{0,i} \\ b_{1,i} \\ b_{2,i} \\ b_{3,i} \end{bmatrix} = \begin{bmatrix} 0x02 & 0x03 & 0x01 & 0x01 \\ 0x01 & 0x02 & 0x03 & 0x01 \\ 0x01 & 0x01 & 0x02 & 0x03 \\ 0x03 & 0x01 & 0x01 & 0x02 \end{bmatrix} \cdot \begin{bmatrix} a_{0,i} \\ a_{1,i} \\ a_{2,i} \\ a_{3,i} \end{bmatrix}$$

This operation actually corresponds to a polynomial multiplication in $\mathbb{F}_{2^8}[x]/(x^4 + 1)$, the ring of polynomials whose coefficient are in $\mathbb{F}_{2^8}$, i.e., they are polynomials themselves. Observe that $x^4 + 1$ is not irreducible in $\mathbb{F}_{2^8}$ and therefore $\mathbb{F}_{2^8}[x]/(x^4 + 1)$ is indeed a ring not a field.

$$a(x) \cdot c(x) = (a_3x^3 + a_2x^2 + a_1x + a_0) \cdot (0x03x^3 + 0x01x^2 + 0x01x + 0x02) (\text{mod } x^4 + 1)$$

We can picture *MixColumns* as

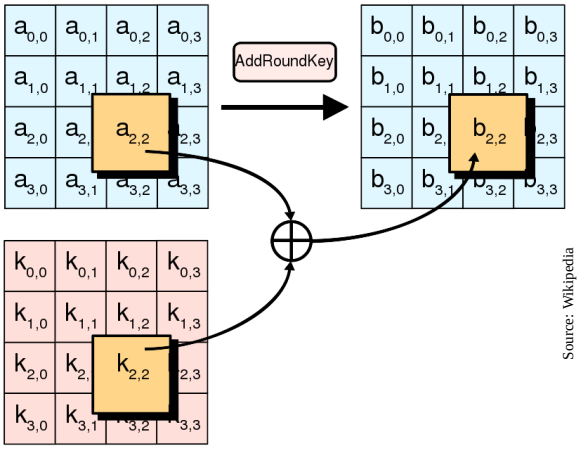


The inverse operation *InverseMixColumns* is then simply the multiplication with $c(x)^{-1}$, which has an inverse in the ring $\mathbb{F}_{2^8}[x]/(x^4+1)$.

4. AddRoundKey: As mentioned earlier, the key size of AES is 128 bits. All the round keys $K_0, \ldots, K_{10}$ derived from the key K are also 128 bits and can therefore be expressed as a 4×4 matrix:

$$K_i = \begin{bmatrix} k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\ k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} \\ k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} \end{bmatrix}$$

Adding the round key is simply the xor-ing of the state matrix and the round key matrix byte by byte: $A \oplus K_i$. The inverse of this operation is obviously the same and we do not need a special operation for the decryption.



30. Key Schedule of AES

The round keys $K_0, \ldots, K_{10}$ are derived from the key K. Since K is 128 bits we can divide it into 4 words of 32 bits each: $K = W_0 \| W_1 \| W_2 \| W_3$, which also corresponds to round key K_0. All subsequent round keys $K_i = W_{4i} \| W_{4i+1} \| W_{4i+2} \| W_{4i+3}$ are then computed using an 8 bit left rotation, an S-Box substitution with the *SubBytes* function, and a scrambling sequence, which is started with a round constant RC_i that is computed in $\mathbb{F}_{2^8}$ by

$$RC_i = x^i (\text{mod } x^8 + x^4 + x^3 + x + 1).$$

The algorithm then looks like this:

KeySchedule(K)

$W_0 \| W_1 \| W_2 \| W_3 := K$
for i := 1 **to** 10 **do**
  $T := W_{4i-1} \lll 8$
  $T :=$ SubBytes(T)
  $T := T \oplus RC_i$
  $W_{4i} := W_{4i-4} \oplus T$
  $W_{4i+1} := W_{4i-3} \oplus W_{4i}$
  $W_{4i+2} := W_{4i-2} \oplus W_{4i+1}$
  $W_{4i+3} := W_{4i-1} \oplus W_{4i+2}$
end

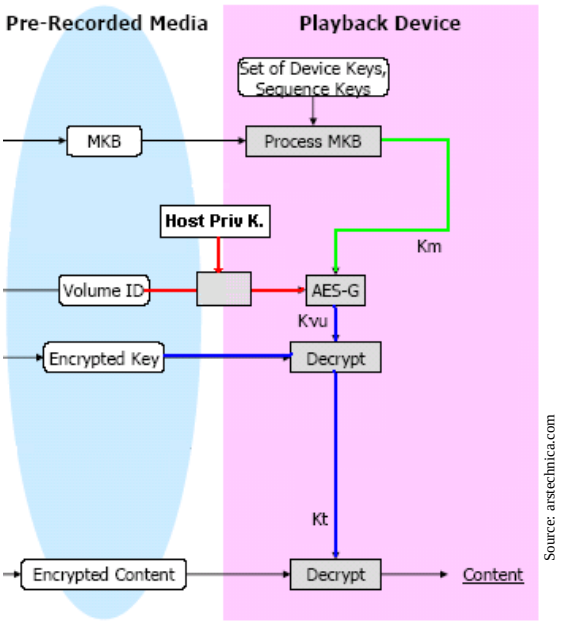
Note that *SubBytes* is applied to the four 8 bit bytes of the 32 bit word T individually.

II.1.4 AACS — Advanced Access Content System

The Advanced Access Content System (AACS) is a Digital Rights Management system for HD-DVD and Blu-Ray Discs developed by a consortium that includes Disney, Intel, Microsoft, Matsushita (Panasonic), Warner Brothers, IBM, Toshiba, and Sony. The general idea of Digital Rights Management (DRM) is to restrict access control to electronic media and playback devices to retain full control by the copyright owner. We first give a brief overview of AACS and then dicuss some of the issues arising from AACS and DRM in general.

31. **Overview** AACS uses encryption, hash functions, and watermarking schemes based on AES. Exact specifications for parts of the technology are published by the AACS licensing authority at `http://www.aacsla.com`. Below is a schematic overview of the mechanism of AACS.

    The content of a disc is encrypted with AES using a collection of *title keys*. The encrypted content, together with the encrypted title keys, a *media key block (MKB)*, and the *unique disc ID* are all stored on the disc.
    An AACS encrypted disc can only be accessed with a fully licensed player. Each licensed player gets a unique set of *device keys* as well as a unique *Host Private Key*. The latter makes it possible that particular playback devices or programmes can be individually revoked.
    The device keys are used to compute a key from the MKB. The host private key is needed to retrieve the disc ID. Both values are then processed by an AES-based one-way function, AES-G. (One-way functions are functions easy to compute, but infeasible to invert. We will discuss one-way functions in more detail later in the lecture.)

    The result of AES-G is then used to decrypt the title keys, which in turn are used to finally decrypt the content.

32. **Security Issues**
    Here are some of the security requirements for AACS implementations to ensure DRM requirements:

    - The content should “not be present on any User-Accessible Bus in analog or unencrypted, compressed form”, because users could possibly record or redirect that content.
    - Implementations must use “encryption, execution of a portion of the implementation in ring zero or supervisor mode (i.e., in kernel mode), and/or embodiment in a secure physical implementation,” to keep encryption keys secret at all times.
    - They must also use “techniques of obfuscation clearly designed to effectively disguise and hamper attempts to discover the approaches used”.

    Thus, video content must travel through the system encrypted and must only interact with authorized components over authorized pathways.
    For example in Windows Vista AACS is implemented via the Protected Video Path (PVP), which stops DRM-restricted content from playing while unsigned software is running in order to prevent the unsigned software from accessing the content. Additionally, PVP can encrypt information during transmission to the monitor or the graphics card, which makes it more difficult to make unauthorized recordings. In other word, you cannot watch a video, while running your own programmes.

33. Legal and Ethical Issues

It is obvious that to fully comply with the DRM requirement specification, AACS has to be deeply embedded into an operating system. This essentially rules out any open source implementation of the AACS as the algorithms to decrypt keys would be easily accessible and therefore security would be compromised. But even if only proprietary implementations are licensed and allowed, any Operating System can (at least theoretically) be emulated by a virtual machine. But running an AACS compliant software player on the VR would give access to the unsecured data streams. This has indeed already been done:

In December 2006 the first software was announced and subsequently published that enables to backup AACS encrypted content. In order to keep the software legal, it cannot actually be used on its own, but an appropriate key must be supplied manually. Since extracting a key without license would be illegal, 128 bit keys were quickly made available throughout the Internet. The AACS licensing agency tried to put a stop to this by suing websites publishing keys as well as by revoking keys (in particular for Windows based software like WinDVD). However, the more they sued, the more keys were published, etc.

The issue remains unsolved to date, in particular since much is a legal grey area. Some of the issues of DRM highlighted by this controversy are:

- The AACS licensing agency argued that publishing the keys is illegal since they would fall under their software patent. However, is it possible to patent single 128-bit numbers? And if so, in which form? Only the hexadecimal representation, or all other representations as well, i.e. decimal, binary, octal, sum of two or several values, etc.? AACS can be used with millions of keys. Are they all patented? Can I be sued if I use one, publish one, etc.

- Under the laws of many countries making one backup of media content is legal. AACS however effectively prevents this.

- AACS protected media can only be played on specially licensed devises, not necessarily on the playback devise of my choice. In other words, although I own the content of the disc (not the Copyright!) I can not play it the way I want to.

  As a comparison: If I buy a bottle of Coke, I am not allowed to reproduce its content, as it is a patented formula. However, no-one can force me to only drink it out of a particular licensed glass.

- The DRM requirements of AACS can only be realised when fully embedded into the operating system. In order not to compromise security (and thus violate the licensing terms) information on the embedding cannot be made public or even given to competitors. However, this amounts essentially to a similar controversy that has led to anti-trust law suits, by companies like Netscape, RealPlayer etc, against Microsoft.

These are just some issues. There are plenty more...

Mathematics 3 – Finite Fields

Finite fields are an important algebraic concepts used in more advanced cryptographic techniques. This section will introduce the very basics, as far as we need them in this lecture. For a more in-depth introduction, see for instance: Lidl & Niederreiter, *Introduction to finite fields and their applications*, Cambridge University Press, 1994.
We first define some basic algebraic notions characterising sets together with binary operations.

Definition 1 ***(Group)*** *Let G be a set and $\circ$ be a binary operation defined on G, i.e., $\circ : G \times G \to G$. We say $(G, \circ)$ is a **group** if $\circ$*

*(i) is **closed**: that is for each $a, b \in G$ we have $a \circ b \in G$.*

*(ii) is **associative**: that is for each $a, b, c \in G$ we have $(a \circ b) \circ c = a \circ (b \circ c)$.*

*(iii) has an **identity element**: that is there exists $e \in G$ s.t. for each $a \in G$ we have $a \circ e = e \circ a = a$.*

*(iv) every element has an **inverse**: that is for every $a \in G$ there is $a^{-1} \in G$ with $a \circ a^{-1} = a^{-1} \circ a = e$.*

*We call $(G, \circ)$ a **commutative group** if in addition $\circ$*

*(v) is **commutative**: that is for each $a, b \in G$ we have $a \circ b = b \circ a$.*

Example: An easy example are the integers with addition $(\mathbb{Z}, +)$. They form a commutative group, since they are clearly closed and associative, the identity element is 0 and each element $a \in \mathbb{Z}$ has an inverse, namely $-a \in \mathbb{Z}$.
On the contrary the integers with times $(Z, \cdot)$ are not a group! Although they are closed under $\cdot$, times is associative, and $1 \in \mathbb{Z}$ is an identity element, not every element has an inverse. For example $2 \in \mathbb{Z}$ would have $\frac{1}{2}$ as inverse, which is not an integer!
However, the relationship between $+$ and $\cdot$ gives rise to the following definition:

Definition 2 ***(Ring)*** *Let R be a set with two binary operations $+$ and $\cdot$, then $(R, +, \cdot)$ is a ring if*

- *$(R, +)$ is a commutative group,*
- *$(R, \cdot)$ is closed, associative and has an identity.*
- *$+$ and $\cdot$ are*
  - *(i) **left distributive**: that is for each $a, b, c \in R$ we have $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$,*
  - *(ii) **right distributive**: that is for each $a, b, c \in R$ we have $(b + c) \cdot a = (b \cdot a) + (c \cdot a)$.*

Example: $(\mathbb{Z}, +, \cdot)$ is a ring, since we can easily verify the distributivity laws.
If the multiplication has inverses as well, we can extend our ring definition to the following:

Definition 3 ***(Field)*** *Let F be a set with two binary operations $+$ and $\cdot$. Let F^* be the set that contains all elements of F except the identity for $+$, i.e. we let $F^* = F \setminus \{0\}$, where 0 is the identity for $+$. Then $(F, +, \cdot)$ is a field if*

- *$(F, +)$ is a commutative group,*
- *$(F^*, \cdot)$ is a commutative group,*
- *$+$ and $\cdot$ are left and right distributive.*

Example: The rational numbers with addition and multiplication form a field: $(\mathbb{Q}, +, \cdot)$. Both operations are obviously closed, associative and commutative. For addition 0 is the identity and for every $a \in \mathbb{Q}$ the additive inverse is $-a \in \mathbb{Q}^*$. Then $\mathbb{Q}^* = \mathbb{Q} \setminus \{0\}$, 1 is the identity for multiplication and every $a \in \mathbb{Q}^*$ has a multiplicative inverse, namely $\frac{1}{a} \in \mathbb{Q}^*$.

Some Finite Examples

In the following we want to restrict ourselves to finite sets. We have already seen some examples of finite sets in Mathematics 1 + 2. For example, the set of all permutations of n elements forms non-commutative(!) group, the symmetric group S_n. Other finite sets are the sets of residue classes modulo some n, $\mathbb{Z}_n$, for which we now want to check what structure they form.

For finite sets one can use a very easy technique to verify the properties of a particular operation, by simply writing down the entire operation in the form of a *multiplication table*. For example we can use the following multiplication tables

$(\mathbb{Z}_2, +)$:

$+$	$[0]_2$	$[1]_2$
$[0]_2$	$[0]_2$	$[1]_2$
$[1]_2$	$[1]_2$	$[0]_2$

$(\mathbb{Z}_2, \cdot)$:

$\cdot$	$[0]_2$	$[1]_2$
$[0]_2$	$[0]_2$	$[0]_2$
$[1]_2$	$[0]_2$	$[1]_2$

$(\mathbb{Z}_3, +)$:

$+$	$[0]_3$	$[1]_3$	$[2]_3$
$[0]_3$	$[0]_3$	$[1]_3$	$[2]_3$
$[1]_3$	$[1]_3$	$[2]_3$	$[0]_3$
$[2]_3$	$[2]_3$	$[0]_3$	$[1]_3$

$(\mathbb{Z}_3, \cdot)$:

$\cdot$	$[0]_3$	$[1]_3$	$[2]_3$
$[0]_3$	$[0]_3$	$[0]_3$	$[0]_3$
$[1]_3$	$[0]_3$	$[1]_3$	$[2]_3$
$[2]_3$	$[0]_3$	$[2]_3$	$[1]_3$

to determine that $(\mathbb{Z}_2, +)$ and $(\mathbb{Z}_3, +)$ are commutative groups. Obviously $(\mathbb{Z}_2, \cdot)$ and $(\mathbb{Z}_3, \cdot)$ are not groups, since neither $[0]_2$ nor $[0]_3$ have a multiplicative inverse. However, if we get rid of the 0 element in both tables, it is easy to see that we get commutative groups for $\mathbb{Z}_2^*$ and $\mathbb{Z}_3^*$:

$(\mathbb{Z}_2^*, \cdot)$:

$\cdot$	$[1]_2$
$[1]_2$	$[1]_2$

$(\mathbb{Z}_3^*, \cdot)$:

$\cdot$	$[1]_3$	$[2]_3$
$[1]_3$	$[1]_3$	$[2]_3$
$[2]_3$	$[2]_3$	$[1]_3$

Since one can also easily check that both distributivity laws hold, we can thus conclude that both $(\mathbb{Z}_2, +, \cdot)$ and $(\mathbb{Z}_3, +, \cdot)$ are fields.

The natural next question is: Do all residue class sets form a field together with addition and multiplication? Let's have a look at $\mathbb{Z}_4$:

$(\mathbb{Z}_4, +)$:

$+$	$[0]_4$	$[1]_4$	$[2]_4$	$[3]_4$
$[0]_4$	$[0]_4$	$[1]_4$	$[2]_4$	$[3]_4$
$[1]_4$	$[1]_4$	$[2]_4$	$[3]_4$	$[0]_4$
$[2]_4$	$[2]_4$	$[3]_4$	$[0]_4$	$[1]_4$
$[3]_4$	$[3]_4$	$[0]_4$	$[1]_4$	$[2]_4$

$(\mathbb{Z}_4^*, \cdot)$:

$\cdot$	$[1]_4$	$[2]_4$	$[3]_4$
$[1]_4$	$[1]_4$	$[2]_4$	$[3]_4$
$[2]_4$	$[2]_4$	$[0]_4$	$[2]_4$
$[3]_4$	$[3]_4$	$[2]_4$	$[1]_4$

We can see, while $(Z_4, +)$ is a commutative group, $(\mathbb{Z}_4^*, \cdot)$ is not a group: It is not even closed, as $[2]_4 \cdot [2]_4 = [0]_4$ and $[0]_4$ is not an element of $\mathbb{Z}_4^*$, and there is also no inverse element for $[2]_4$. Thus $(\mathbb{Z}_4, +, \cdot)$ is a ring ($[1]_4$ is the neutral element for $\cdot$), but not a field.

Indeed one can show the following two theorems:

Theorem 4 *$(\mathbb{Z}_n, +, \cdot)$ is a ring for every $n \geq 2$.*

Theorem 5 *$(\mathbb{Z}_p, +, \cdot)$ is a field if and only if p is a prime number.*

Definition 6 ***(Finite Field)*** *Let $(F, +, \cdot)$ be a field. If F is a finite set with p elements, we call $(F, +, \cdot)$ a* ***finite field*** *or order p and denote it by $\mathbb{F}_p$.*

Example: We can now write $\mathbb{F}_2 = (\mathbb{Z}_2, +, \cdot)$, $\mathbb{F}_3 = (\mathbb{Z}_3, +, \cdot)$, etc. In general we have for every prime number p: $\mathbb{F}_p = (\mathbb{Z}_p, +, \cdot)$.

So far we know that for every prime number p there exists a finite field of that order. In addition one can easily show that this is (up to isomorphism) the only finite field of that order. That is, every finite field of prime order has the structure of $\mathbb{Z}_p$.

Our next question is, are these the only finite fields, or are there any others, i.e. of an order that is not a prime number. In order to answer this question, we have to make a little detour via the theory of polynomials.

Polynomials

I assume that everyone is familiar with polynomials. The following is just to recall some important concepts. While we can define polynomials essentially over any ring, we will restrict ourselves, for now, to polynomials over the integers $\mathbb{Z}$.

Definition 7 ***(Polynomial)*** *We call an expression of the form $a_n x^n + \ldots + a_2 x^2 + a_1 x + a_0$ a **polynomial** in the variable x over $\mathbb{Z}$, if all $a_i \in \mathbb{Z}, i = 0, \ldots, n$ and all exponents $0, \ldots, n$ are non-negative integers. We denote the set of all polynomials in one variable over $\mathbb{Z}$ as $\mathbb{Z}[x]$.*
*We call a summand $a_i x^i$ of a polynomial a **monomial** of **degree** i with **coefficient** a_i.*
*We say a polynomial $p \in \mathbb{Z}[x]$ is of **degree** n if its greatest non-zero monomial is of degree n. We generally write $deg(p) = n$.*

Example: $p(x) = x^4 + 3x^3 + 2x^2 - 10$ is a polynomial of degree 4.

Definition 8 ***(Polynomial Arithmetic)*** *Let $p(x), q(x) \in \mathbb{Z}[x]$ be $p(x) = a_n x^n + \ldots + a_2 x^2 + a_1 x + a_0$ and $q(x) = b_n x^n + \ldots + b_2 x^2 + b_1 x + b_0$. We define addition $+$ and multiplication $\cdot$ as component-wise operations as:*

(i) $p(x) + q(x) = (a_n + b_n)x^n + \ldots + (a_2 + b_2)x^2 + (a_1 + b_1)x + (a_0 + b_0)$

(ii)
$$\begin{aligned} p(x) * q(x) \quad = \quad & (a_n * b_m)x^{(n+m)} \quad + (a_n * b_{m-1})x^{(n+(m-1))} \quad + \ldots \quad + (a_n * b_0)x^n \\ & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\ + \quad & (a_0 * b_m)x^m \qquad + (a_0 * b_{m-1})x^{m-1} \qquad + \ldots \quad + (a_0 * b_0) \end{aligned}$$

Example: Let $p(x) = x^4 + 3x^3 + 2x^2 - 10$ and $q(x) = 2x^3 - 9x^2 + 2x - 3$. Then we have

$$\begin{aligned} p(x) + q(x) &= x^4 + 5x^3 - 7x^2 + 2x - 13 \\ p(x) \cdot q(x) &= 2x^7 - 3x^6 - 21x^5 - 15x^4 - 25x^3 + 84x^2 - 20x + 30. \end{aligned}$$

Recall that $(\mathbb{Z}, +, \cdot)$ forms a ring. Similarly we can show that $(\mathbb{Z}[x], +, \cdot)$ forms a ring with the addition and multiplication over polynomials. We now observe some more parallels between $\mathbb{Z}$ and $\mathbb{Z}[x]$:

$\mathbb{Z}$	$\mathbb{Z}[x]$
Division with Remainder	
Divide 323 by 7 $323 \;/\; 7 = 46$ -28 43 -42 1	Divide $x^3 + 4x^2 + 6x - 1$ by $x^2 + 2x + 1$ $x^3 + 4x^2 + 6x - 1 \;/\; x^2 + 2x + 1 = x + 2$ $-x^3 - 2x^2 - x$ $2x^2 + 5x - 1$ $-2x^2 - 4x - 2$ $x - 3$
In general for every $a, b \in \mathbb{Z}$ with $a \geq b$ we find $s, r \in \mathbb{Z}$ with $\lvert s\rvert < \lvert a\rvert$ and $\lvert r\rvert < \lvert b\rvert$ such that $a = s \cdot b + r$.	In general for every $p(x), q(x) \in \mathbb{Z}[x]$ with $deg(p) \geq deg(q)$ we find $s(x), r(x) \in \mathbb{Z}[x]$ with $deg(s) < deg(p)$ and $deg(r) < deg(q)$ such that $k \cdot p(x) = s(x) \cdot q(x) + r(x)$, with $k \in \mathbb{Z}$. Observe that k guarantees integer division for the coefficients (e.g. $x^2 + 1$ is not divisible by $2x + 1$ in $\mathbb{Z}$, but $4x^2 + 4$ is). Such a k always exists!
Modular Arithmetic	
Recall: $323 \equiv 1 (\mathrm{mod}\ 7)$.	Similarly we can write $x^3 + 4x^2 + 6x - 1 \equiv x - 3 (\mathrm{mod}\ x^2 + 2x + 1)$
The modulo operation divides $\mathbb{Z}$ into a finite number of residue classes, e.g., for mod n: $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$	Given a polynomial $p(x) \in \mathbb{Z}[x]$, the modulo $p(x)$ operation induces residue classes on $\mathbb{Z}[x]$. We denote the set of all residue classes modulo $p(x)$ by $\mathbb{Z}[x]/p(x)$. It contains one residue class for each polynomial in $\mathbb{Z}[x]$ of degree less then $p(x)$. These are, however, infinitely many!.
Irreducibility	
$p \in \mathbb{Z}$ is prime if it is only divisible by 1 and p.	$p(x) \in \mathbb{Z}[x]$ is called *irreducible* if it is only divisible by $p(x)$ and the trivial polynomial $a_0 x^0 = a_0 \in \mathbb{Z}$. E.g., $x^2 + 1$ is irreducible in $\mathbb{Z}[x]$.

Polynomials over Finite Fields

As mentioned earlier we can construct polynomials over arbitrary rings and therefore also fields. We will now look at the residue class construction for polynomials over finite fields. While the following could be done with any finite field of the form $\mathbb{F}_p$, where p is a prime number, we will restrict ourselves to the finite field $\mathbb{F}_2$, in which we are most interested in.

Recall that $\mathbb{F}_2 = (\mathbb{Z}_2, +, \cdot)$, i.e. contains only the elements 0 and 1. We will from now on omit the residue class notation and write 0 and 1 instead of $[0]_2$ and $[1]_2$, respectively!

We now define the polynomial ring over $\mathbb{F}_2$ as $\mathbb{F}_2[x]$. We first have a look at the general polynomial arithmetic in $\mathbb{F}_2[x]$, which works modulo 2, that is we only have 0 and 1 as coefficients and the addition and multiplication of coefficient is performed modulo 2. We observe this with an example:

Example: Let $x^2 + x + 1$ and $x^3 + x^2 + x$ be polynomials over $\mathbb{F}_2[x]$ then we have:

$$\begin{aligned}(x^2 + x + 1) + (x^3 + x^2 + x) &= x^3 + 2x^2 + 2x + 1 &&= x^3 + 1\\ (x^2 + x + 1) \cdot (x^3 + x^2 + x) &= x^5 + 2 * x^4 + 3 * x^3 + 2 * x^2 + x &&= x^5 + x^3 + x\end{aligned}$$

This also means that we do not need negative coefficients, as $x + 1 = -x + 1 = x - 1 = -x - 1$ in $\mathbb{F}_2[x]$.

Modular Arithmetic Let $p(x) = x^4 + x + 1 \in \mathbb{F}_2[x]$. Then we have

$$(x^2 + x + 1) \cdot (x^3 + x^2 + x) = x^5 + x^3 + x \equiv x^3 + x^2 (\mathrm{mod}\ x^4 + x + 1)$$

To verify this we look at the following polynomial division:

$$\begin{array}{llll} x^5 & +x^3 & & +x \quad / \quad x^4 + x + 1 = x \\ \cline{1-3} x^5 & & +x^2 & +x \\ \cline{3-4} & x^3 & +x^2 & \end{array}$$

In a next step we can now define the residue classes for $\mathbb{F}_2[x]/p(x)$. For simplification we take $p(x)$ to be polynomial of degree 2 first. Thus let $p(x) = x^2 + 1 \in \mathbb{F}_2[x]/p(x)$. We then get four residue classes modulo $x^2 + 1$, represented by $[0]$, $[1]$, $[x]$, $[x + 1]$, i.e., all polynomials in $\mathbb{F}_2[x]$ with degree less than 2. We can then construct the following tables for $+$ and $\cdot$:

$(\mathbb{F}_2[x]/(x^2 + 1), +)$:

$+$	$[0]$	$[1]$	$[x]$	$[x+1]$
$[0]$	$[0]$	$[1]$	$[x]$	$[x+1]$
$[1]$	$[1]$	$[0]$	$[x+1]$	$[x]$
$[x]$	$[x]$	$[x+1]$	$[0]$	$[1]$
$[x+1]$	$[x+1]$	$[x]$	$[1]$	$[0]$

$(\mathbb{F}_2[x]/(x^2 + 1), \cdot)$:

$\cdot$	$[0]$	$[1]$	$[x]$	$[x+1]$
$[0]$	$[0]$	$[0]$	$[0]$	$[0]$
$[1]$	$[0]$	$[1]$	$[x]$	$[x+1]$
$[x]$	$[0]$	$[x]$	$[1]$	$[x+1]$
$[x+1]$	$[0]$	$[x+1]$	$[x+1]$	$[0]$

We can see that, even when we delete the $[0]$ lines in the multiplication table, the group axioms will not hold for '$\cdot$', as $[x + 1]$ does not have an inverse element. This can be explained by the fact that $x^2 + 1$ is not an irreducible polynomial in $\mathbb{F}_2[x]$, since it can be factorised into $x^2 + 1 = (x + 1)(x + 1)$. We recall that for those $\mathbb{Z}_n$ where n was a prime number we could construct a finite field. If we replace $x^2 + 1$ by an irreducible polynomial we should therefore also get a finite field. Let's try instead the polynomial $p(x) = x^2 + x + 1$, which is indeed irreducible over $\mathbb{F}_2[x]$.

$(\mathbb{F}_2[x]/(x^2 + x + 1), +)$:

$+$	$[0]$	$[1]$	$[x]$	$[x+1]$
$[0]$	$[0]$	$[1]$	$[x]$	$[x+1]$
$[1]$	$[1]$	$[0]$	$[x+1]$	$[x]$
$[x]$	$[x]$	$[x+1]$	$[0]$	$[1]$
$[x+1]$	$[x+1]$	$[x]$	$[1]$	$[0]$

$(\mathbb{F}_2[x]/(x^2 + x + 1), \cdot)$:

$\cdot$	$[0]$	$[1]$	$[x]$	$[x+1]$
$[0]$	$[0]$	$[0]$	$[0]$	$[0]$
$[1]$	$[0]$	$[1]$	$[x]$	$[x+1]$
$[x]$	$[0]$	$[x]$	$[x+1]$	$[1]$
$[x+1]$	$[0]$	$[x+1]$	$[1]$	$[x]$

The two tables demonstrate that both $(\mathbb{F}_2[x]/(x^2 + x + 1), +)$ and $((\mathbb{F}_2[x]/(x^2 + x + 1))^*, \cdot)$ form commutative groups. [Recall that $(\mathbb{F}_2[x]/(x^2 + x + 1))^* = \mathbb{F}_2[x]/(x^2 + x + 1) \setminus \{[0]\}$.] We can also show that the two distributivity laws hold and that therefore $((\mathbb{F}_2[x]/(x^2 + x + 1))^*, +, \cdot)$ is a finite field of order 4.

This construction demonstrates that for every irreducible polynomial $p(x) \in \mathbb{F}_2[x]$ of degree n, $\mathbb{F}_2[x]/p(x)$ yields a finite field of order 2^n, independent of the concrete choice of $p(x)$. The more general result is:

Theorem 9 *For every prime p and every positive integer n there exists one finite field of order $\mathbb{F}_{p^n}$.*

Finite Fields as Binary Operations

What does all this have to do with computer science? Recall that we restricted ourselves to polynomials in $p(x) \in \mathbb{F}_2[x]$. This means each monomial in $p(x)$ has as coefficient either 1 or 0, i.e. the coefficients are binary. We can therefore straightforwardly translate polynomials of degree d into bit strings of length $d+1$ by just taking the coefficients of each monomial.

Example: Consider the translation of a polynomial of degree 7 into 8 bits:

$$\begin{array}{cccccccc} x^7 & +x^6 & & +x^4 & +x^3 & & & +1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{array}$$

We now can use the operations on the finite field $\mathbb{F}_{2^n}$ to define operations on bit strings of length $n+1$. For brevity we will take our examples from $\mathbb{F}_8 = \mathbb{F}_2[x]/p(x)$ with $p(x) = x^3 + x + 1$ as irreducible polynomial.

We first observe that addition on $\mathbb{F}_{2^n}$ is the same as the xor operation $\oplus$ on bits.

Example:
$$\begin{array}{ccccc} (x^2+x+1) & + & (x^2+1) & = & x \\ 111 & \oplus & 101 & = & 010 \end{array}$$

While addition does not give us a new operation, we can use multiplication on $\mathbb{F}_8$ to define a new bitwise operation $\otimes$. First here is the multiplication table for $\mathbb{F}_8^*$:

$(\mathbb{F}_8^*, \cdot)$	$[1]$	$[x]$	$[x+1]$	$[x^2]$	$[x^2+1]$	$[x^2+x]$	$[x^2+x+1]$
$[1]$	$[1]$	$[x]$	$[x+1]$	$[x^2]$	$[x^2+1]$	$[x^2+x]$	$[x^2+x+1]$
$[x]$	$[x]$	$[x^2]$	$[x^2+x]$	$[x+1]$	$[1]$	$[x^2+x+1]$	$[x^2+1]$
$[x+1]$	$[x+1]$	$[x^2+x]$	$[x^2+1]$	$[x^2+x+1]$	$[x^2]$	$[1]$	$[x]$
$[x^2]$	$[x^2]$	$[x+1]$	$[x^2+x+1]$	$[x^2+x]$	$[x]$	$[x^2+1]$	$[1]$
$[x^2+1]$	$[x^2+1]$	$[1]$	$[x^2]$	$[x]$	$[x^2+x+1]$	$[x+1]$	$[x^2+x]$
$[x^2+x]$	$[x^2+x]$	$[x^2+x+1]$	$[1]$	$[x^2+1]$	$[x+1]$	$[x]$	$[x^2]$
$[x^2+x+1]$	$[x^2+x+1]$	$[x^2+1]$	$[x]$	$[1]$	$[x^2+x]$	$[x^2]$	$[x+1]$

The operation $b_1 \otimes b_2$ on 3-bit strings b_1, b_2 is then defined taking the two polynomials corresponding to b_1 and b_2, respectively, multiplying them according to the above multiplication table and transforming the result again into a 3-bit string.

Example:
$$\begin{array}{ccccc} (x^2+x+1) & \cdot & (x^2+1) & \equiv & x^2+x \pmod{x^3+x+1} \\ 111 & \otimes & 101 & = & 110 \end{array}$$

Observe that the choice of irreducible polynomial really matters in the definition of $\otimes$. For instance, if our choice of irreducible polynomial were x^3+x^2+1 then the example would look like this:

Example:
$$\begin{array}{ccccc} (x^2+x+1) & \cdot & (x^2+1) & \equiv & 1 \pmod{x^3+x^2+1} \\ 111 & \otimes & 101 & = & 001 \end{array}$$

Finally we consider another two, more complex examples, where again the choice of irreducible polynomial matters:

Example:

$$\begin{array}{ccccc} (x^6+1) & \cdot & (x^4+1) & \equiv & x^5+x^4+x^3+x^2+1 \pmod{x^8+x^4+x^3+x+1} \\ 01000001 & \otimes & 00010001 & = & 00111101 \end{array}$$

$$\begin{array}{ccccc} (x^6+1) & \cdot & (x^4+1) & \equiv & x^5+x^2+1 \pmod{x^8+x^4+x^3+x^2+1} \\ 01000001 & \otimes & 00010001 & = & 00100101 \end{array}$$

Observe that the bit strings are of length 8, which is a multiple of 4. We can therefore express the above multiplications in terms of hexadecimal numbers, indicated by prefix $0x$:

$$0x41 \otimes 0x11 = 0x3\text{D} \qquad \text{and} \qquad 0x41 \otimes 0x11 = 0x25$$

Mathematics 4 – Matrix Arithmetic

I assume that everyone is familiar with matrices and basic operations on them. This handout should serve as a reminder.
Recall that a matrix is a rectangular array of elements. Abstractly a $m \times n$ matrix can be displayed as:

$$A = \begin{bmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & . & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{bmatrix}$$

We will refer to a particular element in the i's row and j's column as $A[i,j] = a_{i,j}$.

Example: $A = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 7 \\ 4 & 9 & 2 \\ 6 & 0 & 5 \end{bmatrix}$ is a 4×3 matrix. The element $a_{2,3}$ is 7.

The matrix $R = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{bmatrix}$ is a 1×9 matrix, or 9-element row vector.

Matrix Addition

Given $m \times n$ matrices A and B, their sum $A + B$ is the $m \times n$ matrix computed by adding corresponding elements (i.e. $(A + B)[i,j] = A[i,j] + B[i,j]$).

Example: $\begin{bmatrix} 1 & 3 \\ 1 & 0 \\ 1 & 2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 7 & 5 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 1+0 & 3+0 \\ 1+7 & 0+5 \\ 1+2 & 2+1 \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ 8 & 5 \\ 3 & 3 \end{bmatrix}$

Matrix Multiplication

Multiplication of two matrices is well-defined only if the number of columns of the left matrix is the same as the number of rows of the right matrix. If A is an $m \times n$ matrix and B is an $n \times p$ matrix, then their matrix product AB is the $m \times p$ matrix (m rows, p columns) given by:

$$(AB)[i,j] = A[i,1]B[1,j] + A[i,2]B[2,j] + ... + A[i,n]B[n,j] \quad \text{for each pair } i \text{ and } j.$$

Example: $\begin{bmatrix} 1 & 0 & 2 \\ -1 & 3 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 & 1 \\ 2 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} (1 \cdot 3 + 0 \cdot 2 + 2 \cdot 1) & (1 \cdot 1 + 0 \cdot 1 + 2 \cdot 0) \\ (-1 \cdot 3 + 3 \cdot 2 + 1 \cdot 1) & (-1 \cdot 1 + 3 \cdot 1 + 1 \cdot 0) \end{bmatrix} = \begin{bmatrix} 5 & 1 \\ 4 & 2 \end{bmatrix}$

Matrix Multiplication over $\mathbb{F}_2$

So far we have implicitly assumed that our matrices are defined over the integers. However, we can likewise define matrices with elements from $\mathbb{F}_2$. Addition and multiplication are then similarly defined as above with the exception that the operations on the single components are performed modulo 2.

Example: Addition: $\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

Example: Multiplication: $\begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1+1+1 \\ 1+1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$

Similarly, we can define matrices and matrix arithmetic over other fields, in particular over all finite fields of the form $\mathbb{F}_{2^n}$.

Cryptography Glossary 3

AACS	Short for Advanced Access Content System	26
AddRoundKey	The round key addition operation used during AES encryption and decryption.	24
Advanced Access Content System	A Digital Rights Management system for HD-DVDs and Blue-Ray Discs	26
AES	The advanced encryption standard that is the successor of DES.	22
Digital Rights Management	Access control technologies that limit usage of digital media or devices.	26
DRM	Short for Digital Rights Management	26
InverseMixColumns	A column manipulation operations used during AES decryption. Inverse of MixColumns.	24
InverseShiftRows	A byte permutation used during AES decryption. Inverse of Shift Rows.	24
InverseSubBytes	An S-Box like substitution algorithm used during AES decryption. The inverse of SubBytes.	23
MixColumns	A column manipulation operations used during AES encryption.	24
Rijndael	The original name of AES.	22
ShiftRows	A byte permutation used during AES encryption.	24
SubBytes	An S-Box like substitution operation used during AES encryption.	23