🔑 النقاط المهمة في السلايد (اللي في الامتحان)

1️⃣ AES restricts it to

> يعني: رغم إن **Rijndael** كان مرن، **AES قفّل الاختيارات**.

احفظ كده بالحرف تقريبًا:

- `**Block Size` = 128 bits فقط**
  ❌ مفيش 64 ولا 256
- **Key Sizes = 128 / 192 / 256 bits**
  وبيسمّوهم:
  - AES-128
  - AES-192
  - AES-256

📌 سؤال مشهور:

> *Does AES support variable block sizes?*
> ❌ **No, only 128 bits**

2️⃣ An iterative rather than Feistel cipher

دي نقطة مفاهيمية مهمّة جدًا.

- AES **مش Feistel**
- AES **Iterative cipher**

يعني إيه؟

- في **Feistel**:
  → البيانات بتتقسم نصين (Left / Right)
- في **AES**:
  ✅ **بيشتغل على البلوك كله مرة واحدة**

👉 الجملة المهمة اللي تتحفظ:

> **AES operates on the entire data block in every round**

3️⃣ Byte operations – Easy to implement in software

دي سبب اختيار AES عمليًا 👇

- AES بيعتمد على **Byte-level operations**
  - مش محتاج عمليات معقدة
  - علشان كده:
    - سريع
    - مناسب للـ software
    - شائع جدًا في التطبيقات

📌 ممكن تيجي كـ:

> *Why is AES efficient in software?*
> ✔ Because it uses **byte-oriented operations**

🧠 ملخص السلايد في 5 سطور (تحفظه)

- AES uses **128-bit block size only**
- Supported key sizes: **128, 192, 256 bits**
- AES is an **iterative cipher**
- It operates on the **entire data block each round**
- Uses **byte-oriented operations**, easy in software

🔐 الفكرة العامة لـ AES Encryption Process

1️⃣ المدخلات (Inputs)

- **Plaintext**
  → 16 bytes = **128 bits**
- **Key**
  → طولها ممكن يكون:
  128 / 192 / 256 bits

📌 مهما كان طول الـ key
👉 **الـ block دايمًا 128 bits**

2️⃣ Input State

- الـ plaintext بيتحوّل لمصفوفة
  4 × 4 bytes
- اسمها: **State**

> كل العمليات في AES بتحصل على الـ State دي

3️⃣ Initial Transformation

- أول خطوة قبل أي rounds
- اسمها:
  AddRoundKey 👉
- يعني:
  - وأول Round Key
  - XOR بين الـ State

📌 دي بتحصل **مرة واحدة بس في البداية**

4️⃣ Rounds (قلب AES ❤️)

عدد الـ rounds بيعتمد على طول الـ key:

Key Size	Number of Rounds
128 bit	10
192 bit	12
256 bit	14

5️⃣ Round 1 → Round N−1

كل Round فيهم فيه **4 transformations**:

1. **SubBytes**
2. **ShiftRows**
3. **MixColumns**
4. **AddRoundKey**

📌 دول بيتكرروا في كل round
📌 والدكتورة غالبًا بتحب تسأل:

> *How many transformations in a round?*
> ✔️ **Four**

6️⃣ Final Round (Round N)

⚠️ مختلف شوية:

- فيه **3 transformations فقط**
- ❌ **بتتشال MixColumns**

يعني:

1. SubBytes
2. ShiftRows
3. AddRoundKey

📌 دي نقطة امتحانية مهمّة جدًا 🔥

7️⃣ Output

- في الآخر يطلع:
  👉 **Ciphertext**
- 16 bytes = **128 bits**

🔑 Key Expansion (الجزء اللي على اليمين)

- من الـ **Main Key**
- بنولّد:
  - **Round keys**
- كل Round ليه **key مختلفة**
- كلهم طولهم:
  👉 **16 bytes**

🧠 ملخص سريع تحفظه

- AES encrypts **128-bit blocks**
- Plaintext → State (4×4 bytes)
- Initial step: **AddRoundKey**
- Each round uses a **round key**
- Normal rounds = **4 transformations**
- Final round = **3 transformations (no MixColumns)**
- Output = **128-bit ciphertext**

🧮 أولًا: إزاي نحسب عدد الـ Rounds في AES؟

القانون العام (للفهم بس 👇):

> **Nr = 6 + max(Nb , Nk)**

معاني الرموز:

- **Nb** = عدد كلمات 32-bit في الـ block
  - AES block = 128 bits
  - 128 ÷ 32 = **4**
    👉 Nb = 4 (ثابت دايمًا)
- **Nk** = عدد كلمات 32-bit في الـ key
  - Key 128 → 4
  - Key 192 → 6
  - Key 256 → 8

الحساب (للفهم فقط)

- AES-128:
  Nr = 6 + max(4,4) = **10**
- AES-192:
  Nr = 6 + max(4,6) = **12**
- AES-256:
  Nr = 6 + max(4,8) = **14**

📌 **لكن في الامتحان؟**

❌ متحسبش
✅ **احفظهم مباشرة**

✅ اللي تحفظه صمًّا (مهم جدًا)

- **AES-128 → 10 rounds**
- **AES-192 → 12 rounds**
- **AES-256 → 14 rounds**

لو جا سؤال:

> *How many rounds in AES-128?*
> ✔ 10

🖼️ ثانيًا: حفظ رسمة التشفير (اللي على اليمين)

احفظها كـ **3 مراحل واضحة** 👇

1️⃣ قبل الـ rounds

Plaintext
⬇️
AddRoundKey (مرة واحدة بس)

2️⃣ الـ Rounds العادية (× N−1)

كل round فيهم فيه **4 خطوات**:

1. SubBytes
2. ShiftRows
3. MixColumns
4. AddRoundKey

📌 دول بيتكرروا **N−1 مرة**

3️⃣ Last Round (آخر Round)

⚠️ مختلفة:

1. SubBytes
2. ShiftRows
3. AddRoundKey

❌ **مفيش MixColumns**

⬇️
Ciphertext

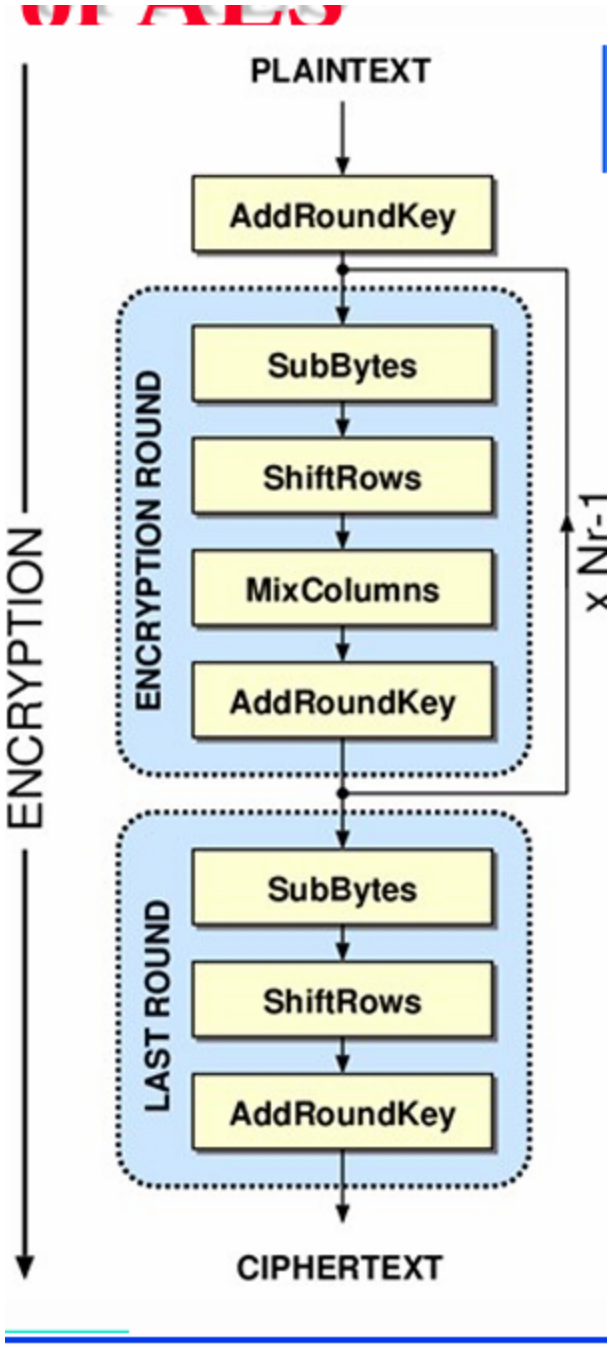
🧠 طريقة حفظ سريعة (هتفيدك جدًا)

الجملة الذهبية:

> **All rounds have 4 steps except the last one (no MixColumns)**

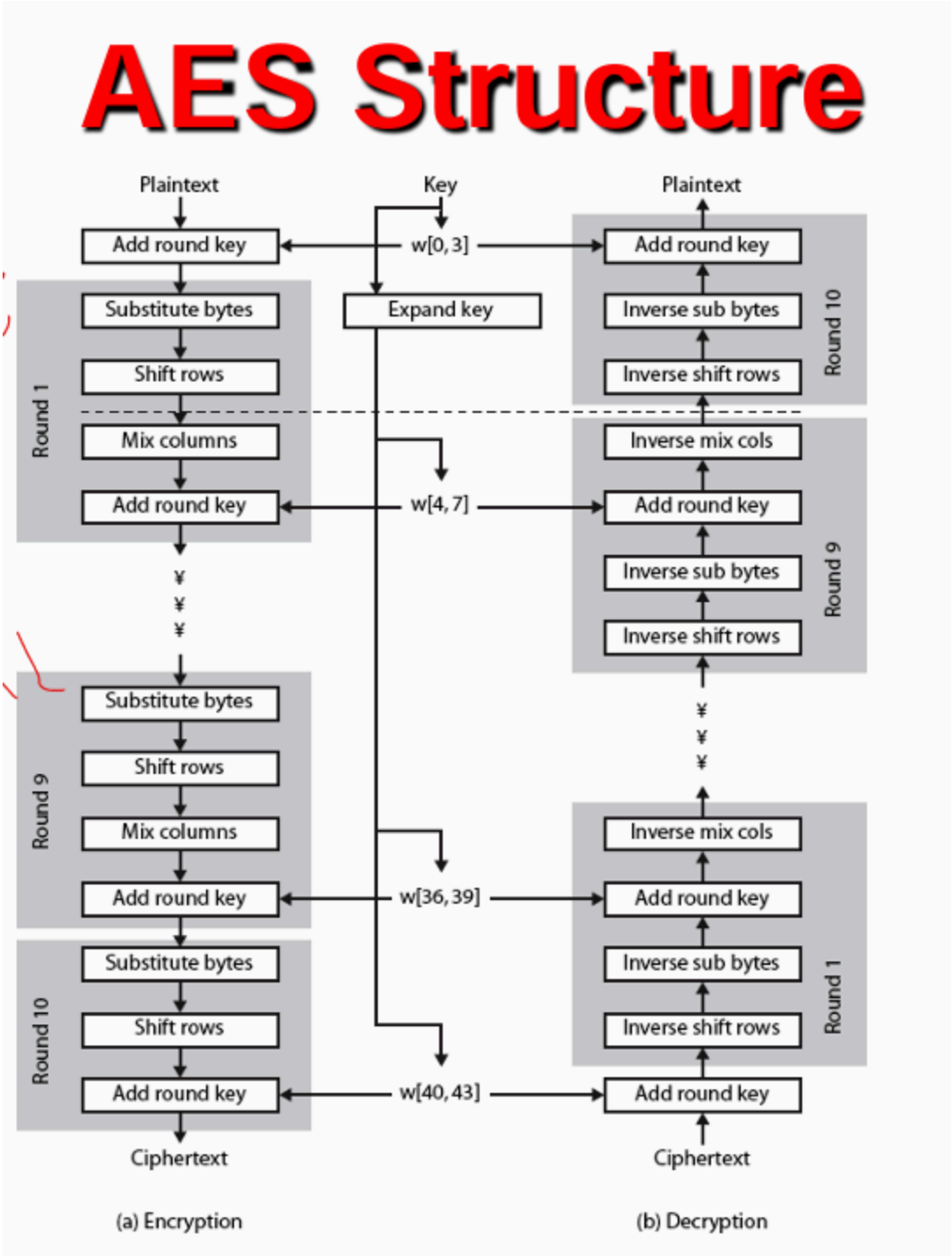
✍️ ملخص السلايد في سطرين

- Number of rounds depends on **key size**
- Final round **does not include MixColumns**



ملخص السلايد في 6 سطور تحفظهم

- AES works on a **4×4 byte state**
- Key is expanded into **round keys**
- Each round applies **four transformations**
- Final round has **no MixColumns**
- Uses **XOR and byte substitution**
- Fast in software implementation



SubBytes (Substitute Bytes) – الفكرة العامة

يعني إيه SubBytes؟

> **Each byte in the state is replaced by another byte using an S-box**

يعني:

- كل **Byte لوحده**
- يتبدّل بـ Byte تاني
- باستخدام **S-box**

📌 مفيش XOR هنا
📌 كل byte مستقل

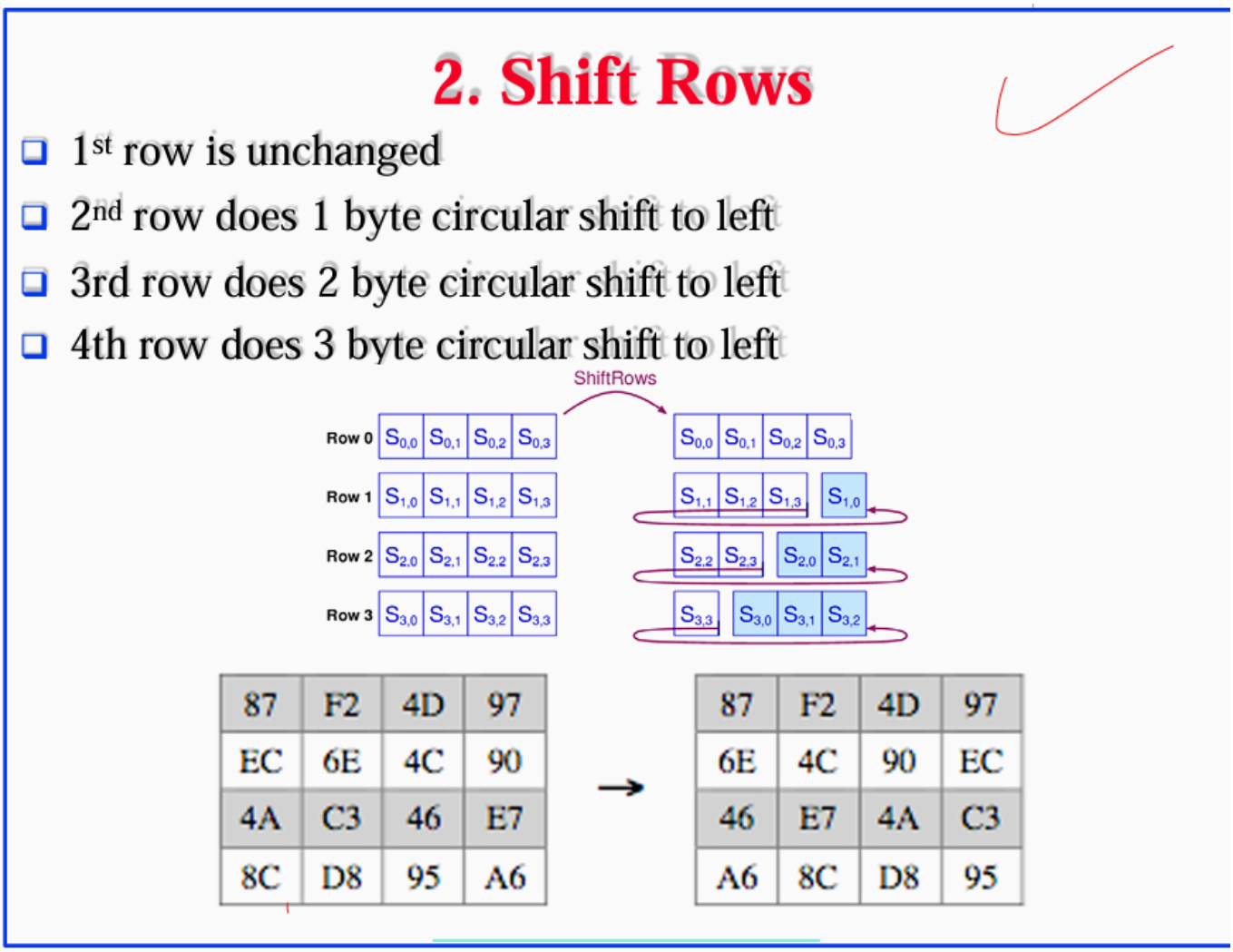
📦 الـ S-box

- جدول حجمه:

> **16 × 16**

- فيه:

> **256 value (ممكن لكل byte)**



- MixColumns transforms each column independently
- Based on matrix multiplication
- Done in GF(2^8)
- Uses fixed polynomial m(x)

- Not applied in the final round

ملخص AddRoundKey في 5 سطور

- XOR state with 128-bit round key
- Byte-by-byte operation
- Uses round keys from key expansion
- Applied in every round
- Only AES step that uses the key

> *What is the purpose of key expansion in AES?*
> ✔ To generate **round keys** from the original key.

🔓 AES Decryption – الفكرة العامة

1️⃣ هل decryption نفس encryption؟

❌ لا، مش **identical**

> AES decryption is **not identical** to encryption
> because the steps are done **in reverse order**

2️⃣ طب ليه بنقول عليهم "متكافئين"؟

رغم إن الخطوات معكوسة:

- نقدر نعرّف:

> **Equivalent inverse cipher**

يعني:

- نفس عدد الـ rounds
- نفس الهيكل العام
- لكن:
  - نستخدم **inverse operations**

🔄 Inverse Operations (من غير تفاصيل)

Encryption	Decryption
SubBytes	**InvSubBytes**
ShiftRows	**InvShiftRows**
MixColumns	**InvMixColumns**
AddRoundKey	AddRoundKey

📌 AddRoundKey:

- **نفس العملية**
- لأن XOR عكس نفسه

🔑 Key Schedule في Decryption

- بيستخدم:

> **Different key schedule**

- يعني:
  - round keys
  - بس بترتيب مختلف (معكوس)

❌ مش مطلوب تفاصيل

🔀 ليه التبديل ده ينفع؟

السلايد قالت نقطة نظرية:

- النتيجة **لا تتغيّر** لو:
  - بدّلنا ترتيب:
    - SubBytes مع ShiftRows
    - MixColumns مع AddRoundKey (بعد tweak)

📌 دي فكرة رياضية، **مش مطلوبة شرح**

🧠 اللي يتحفظ للامتحان

- AES decryption is **not identical** to encryption
- Steps are done in **reverse order**
- Uses **inverse operations**
- Uses a **different key schedule**
- Same structure and number of rounds

✍️ ملخص في 5 سطور

- Decryption reverses encryption steps
- Uses inverse transformations
- AddRoundKey is unchanged
- Round keys are used in reverse order
- Final result recovers plaintext

1️⃣ Block & Key & Rounds

- AES بيشفّر:
  - **Block size = 128 bits**
- باستخدام Keys:
- **128 / 192 / 256 bits**
- وعدد Rounds:
  - **10 / 12 / 14** على الترتيب

> 🔑 128 → 10
> 🔑 192 → 12
> 🔑 256 → 14

2️⃣ Not a Feistel Cipher

- AES مش **Feistel**
- يعني:

> **All 128 bits are encrypted together**

❌ مفيش Left / Right halves

✔ كامل block

3️⃣ Steps in Each Round

كل Round فيه **4 خطوات ثابتة**:

1. SubBytes
2. ShiftRows
3. MixColumns
4. AddRoundKey

📌 دي أكتر نقطة تتسأل

4️⃣ Last Round

- آخر Round:
  - **3 steps فقط**
  - ❌ **No MixColumns**

> SubBytes
> ShiftRows
> AddRoundKey

🔥 نقطة امتحانية مباشرة

5️⃣ Decryption

- Decryption:
  - **مش نفس encryption**
  - بتستخدم:

> **Inverse steps**

📌 زي AES مش زي DES

🧠 ملخص ذهبي في 3 سطور (لو الوقت زنقك)

- AES uses 128-bit blocks and 10/12/14 rounds
- Each round has 4 steps, last round has no MixColumns
- Decryption uses inverse operations