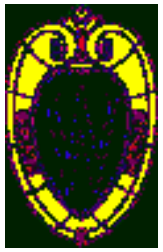


CS/ECE/Math 435
Introduction to Cryptography

Professor Chris DeMarco
Department of Electrical & Computer Engineering
University of Wisconsin-Madison

Spring Semester 2021

Recap : "Ingredients" of AES thus far -

- BS : $GF(2^8) \to GF(2^8)$, "BS step" "core" function, mixing an inverse in $GF(2^8)$ with a matrix multiply (matrix "A") and constant vector add ("b").

- Add Round Key - ARK Step

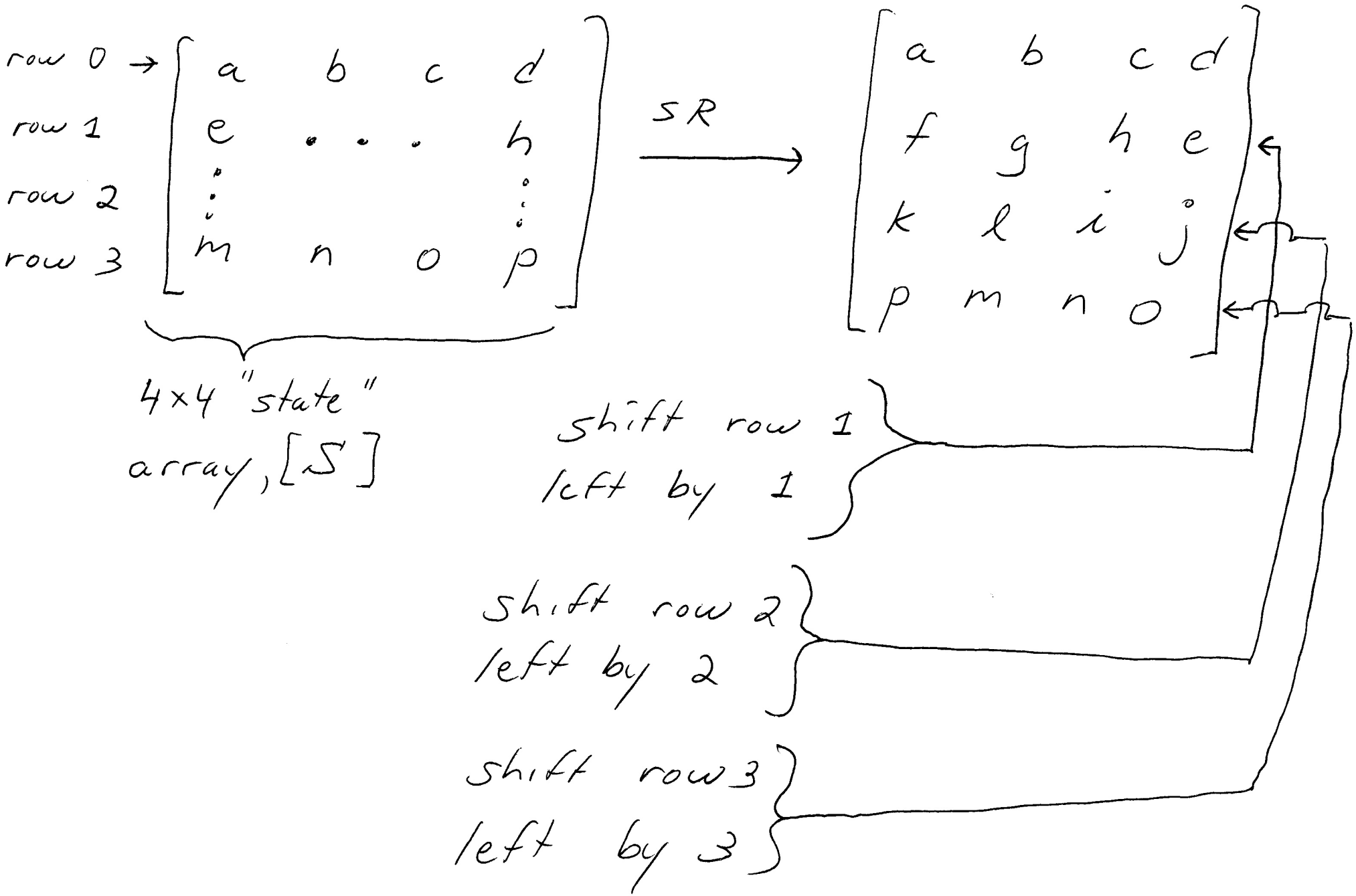
  Recognizing that encryption in AES uses 11 "rounds," one generates 10 new keys from original key k_0

Clever recycling of BS(·) functions to generate intermediate subkeys.

Shift Rows Ingredient

Recall that original input plaintext block, 128-bits, is organized as 4×4 array of 8-bit elements. Each intermediate ciphertext (the "state") is similarly organized.

The SR operator is



The Mix Columns Ingredient

Here we perform a polynomial (i.e. treat elements as $\in GF(2^8)$) matrix multiply:

$$MC([S]) = \begin{bmatrix} X & (X+1) & 1 & 1 \\ 1 & X & (X+1) & 1 \\ 1 & 1 & X & (X+1) \\ (X+1) & 1 & 1 & X \end{bmatrix} [S]$$

↑

again, 4x4 matrix/array with elements in $GF(2^8)$

Overall Block Encryption Operation

for AES: $\underline{X}$ = 128-bit input plaintext

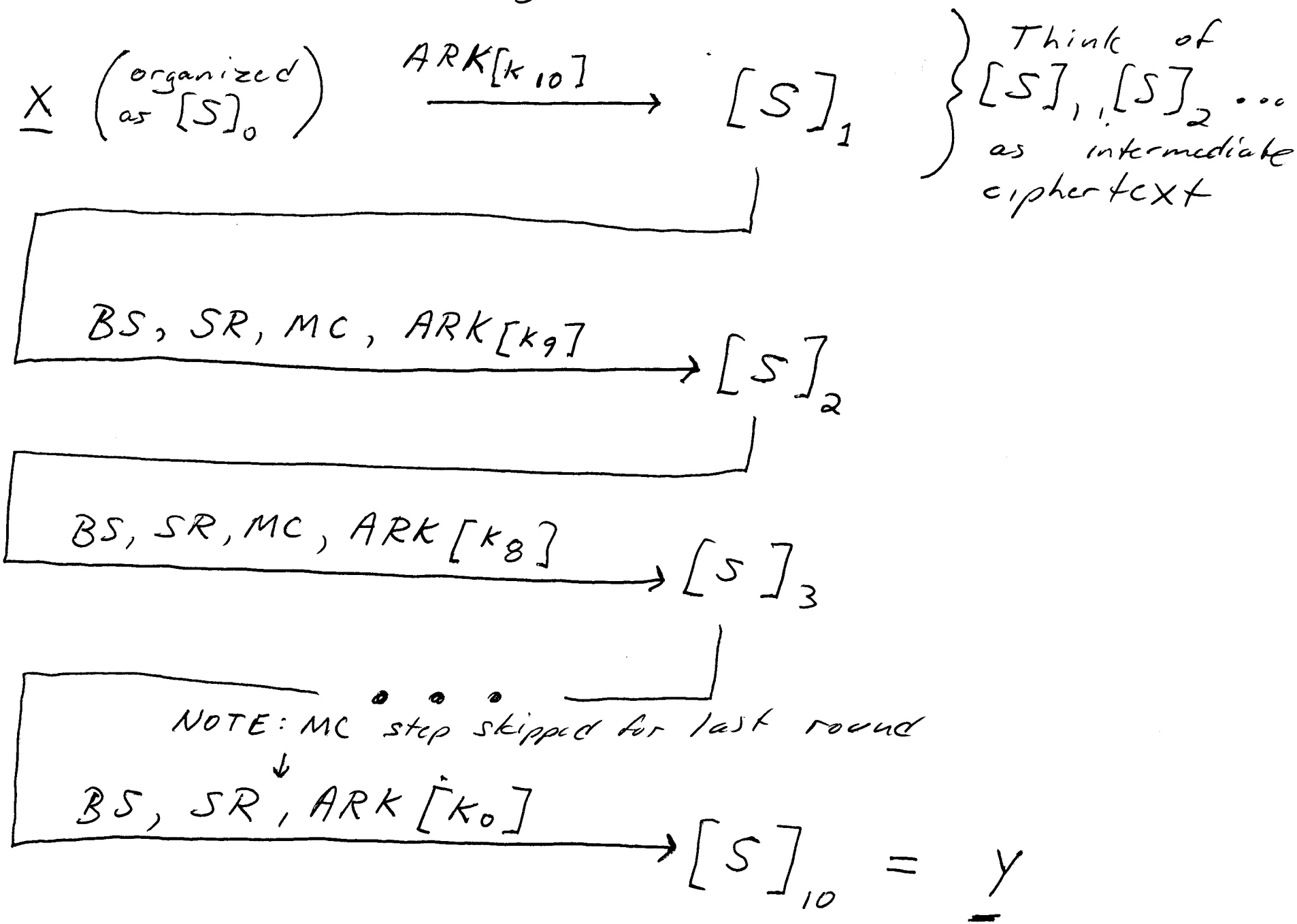
$\underline{Y}$ = 128-bit output ciphertext

$\underline{X}$, and intermediate ciphertext results are organized in data structures composed of 4×4 arrays with 8-bit elements, denoted $[S]$

Eleven "rounds" of operations employing BS, ARK, SR, and MC steps. Each round uses a distinct $\underline{k}$; ($\underline{k}$: 128 bits, then decomposed into four $\underline{c}$ columns)

Caution on key indexing notation:
we "use" keys in reverse order,
relative to their index "i."

Overall flow diagram for AES:



All encryption schemes examined thus far in 435 share two design features:

- Encryption and decryption functions are parameterized by a supplementary input, known as the key. This key must be securely communicated between encrypting entity and authorized decryptor(s). i.e. both encryption and decryption depend on secret key.

- Encryption function $e_k(\cdot)$ and decryption function $d_k(\cdot)$ employ very similar structure (e.g., in Hill, encryption and decryption both employ matrix multiply operation).

  And therefore, encryption and decryption have same order of computational cost.

Classes of encryption systems studied thus far in 435 are therefore known as

"Private Key" Cryptographic Systems

and/or

"Symmetric" Cryptographic Systems

here indicating symmetry between encryption step(s) and decryption step(s)

In contrast to systems studied thus far in 435, next topic (Bach notes section 35) treats

"Public key" Cryptographic Systems

also known as

"Asymmetric" Cryptographic Systems

encryption and "decryption" will no longer be "symmetric" ⇒ the decryption function is allowed to have a different structure and key than the encryption function.

Early published work circa 1978:

Ron Rivest, Adi Shamir, Leonard Adleman

$\Rightarrow$ RSA Public key System

(similar concepts developed in parallel in the UK, but were classified).

Basic Structure of RSA system

Operations will be defined on elements of $\mathbb{Z}_N$. However, unlike our use of $\mathbb{Z}_N$ early in the semester, N typically very large,

and elements of $\mathbb{Z}_N$ are not associated with a natural language alphabet.

Instead, we assume data to be encrypted is a binary file of ℓ-bits. Choose largest integer k such that $2^k \leq N$, and divide length ℓ file into m blocks of length k. Then each length k block is a binary string $[x_{k-1}, x_{k-2}, x_{k-3}, \ldots x_0]$.

Integer representation of a length k block is simply

$$p = x_{k-1} \cdot 2^{k-1} + x_{k-2} \cdot 2^{k-2} \cdots + x_0 \cdot 2^0$$

and $p \in \mathbb{Z}_N$.

So original binary file of m blocks, each composed of k bits, is equivalent to an m-dimensional vector in $\mathbb{Z}_N^m$.

RSA operates on elements of $\mathbb{Z}_N$ one-at-a-time (i.e. it is a monoalphabetic cipher, $\mathbb{Z}_N$ to $\mathbb{Z}_N$).

RSA Initialization:

User selects two distinct prime numbers, p, q. As we will see, security of scheme is enhanced by choice of "large" prime numbers.

Then set $N = p \cdot q$

(observe that N can then be very large $\Rightarrow$ much larger than examples of $\mathbb{Z}_N$ earlier).

Finally, user must identify positive integers $d, e \leq N$ such that $\mod(d \cdot e, (p-1)\cdot(q-1)) = 1$

i.e., d and e are pair of multiplicative inverses with respect to set $\mathbb{Z}_M$, where $M = (p-1)(q-1)$

(recall $N = p \cdot q$).

Note that in our earlier use of monoalphabetic ciphers on $\mathbb{Z}_N$ in 435, we assumed "everyone" knew the set $\mathbb{Z}_N$, because it was associated with a natural language alphabet $\Rightarrow$ value of N known a priori.

Here in RSA, value of N is a <u>user selected choice</u>

$\Rightarrow$ it must be specified as part of the key(s).

Encryption Function: $E : \mathbb{Z}_N \to \mathbb{Z}_N$

The encryption function has an associated Public Key specified by the pair (N, e).

Then $E_{(N,e)}(x) = \mathrm{mod}(x^e, N)$

(N, e): key parameters — we'll soon get lazy, and stop writing these as explicit subscripts

Decryption Function: $D: \mathbb{Z}^N \to \mathbb{Z}^N$

Has associated "private key" or "secret key" (N, d).

(aside: really, once public key for encryption is broadcast, d is the only secret part here).

$$D_{(N,d)}(y) = \mathrm{mod}(y^d, N)$$