# **Cryptography**

AES is now currently being used as the standard block cipher algorithm which has replace the Data Encryption Standards (DES). In 1997 National Institute of Standards and Technology announced for anew cipher algorithms which could replace the DES. An overview summary

of the requirements made by the NIST for the AES are as follows:

* Symmetric-key cipher
* Block cipher
* Support for 128-bit block sizes
* Support for 128-, 192-, and 256-bit key lengths

In October 2000, the Rijndel Algorithm was chosen as the new standard encryption algorithm. The operation of the AES algorithm is shown in the below flow chart. The steps of encryption use a key which converts which convert the input data into an unreadable text (Cipher Text). In the decryption the same key which was used in the encryption is used to convert the cipher text into its original form.

**Plaintext 00112233445566778899aabbccddeeff**

**Encryption:**

**Key 000102030405060708090a0b0c0d0e0f**

**Cipher text 69a2ff1d9df52d1cv1d3f68dcd1c84a1**

**Decryption:**

**Key 000102030405060708090a0b0c0d0e0f**

**Plaintext 00112233445566778899aabbccddeeff**

The process of encryption and decryption is quite simple but some of the detailed cryptographic is required to understand what is happening in the whole process. Algorithms such as AES are called product ciphers. For such type of ciphers encryption is done in rounds, in which each round processing is done using the same logic. Furthermore, many of these ciphers including the AES can change the key use in the encryption at each round. The strength of encryption done by the product cipher can be improved by increasing the number of rounds used to process the data. AES standard determines that the number of rounds is specified by the length of the cipher key, shown below:

|  |  |
| --- | --- |
| **Key Length** | **Number Of Rounds** |
| 128 | 10 |
| 192 | 12 |
| 254 | 14 |

**Example: AES Implementation On GPU**

As we have seen how the encryption and decryption of AES occurs. Now, let’s see how its implementation looks like in as a vertex program.

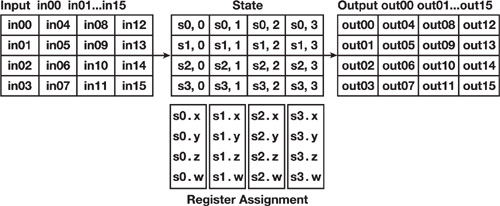
**Working:**

|  |
| --- |
| **!!NVvp4.0**  **/\***  **\* aes.vpt -- AES encryption and decryption**  **\* Author Yamanouchi\_Takeshi@sega.co.jp**  **\*/**  **// input, output**  **ATTRIB state\_in = vertex.attrib[0];**  **OUTPUT state\_out = result.texcoord[0];**  **// macros**  **INT TEMP \_tmp0, \_tmp1; // macro work**  **#define ROT8(\_arg) \**  **SHL.U \_tmp0, \_arg, 8; \**  **SHR.U \_arg, \_arg, 24; \**  **OR \_arg, \_arg, \_tmp0** |

Here we have expanded the cipher key by using the CPU and storing the GPU program local parameter.

**Program Input/output and The State:**

The AES encryption works over a two dimensional array of bytes which is called ‘The State’. We have divided our data into 16 bytes of sequential blocks and unpacked it into 4x4 arrays that we had pushed into the GPU’s register. During the output step, we have packed the 4x4 array back into sequential block of 16 bytes as shown in the Figure below:



In the initialization stage, we have done an AddroundKey Operation which is also a XOR operation on the state by the round key. In our case we have performed unpacking and round key addition together. The round for AES operation consist of four operations:

### SubBytes Operation

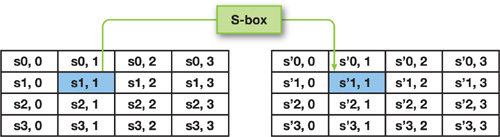
### ShiftRows Operation

### MixColumns operation

### AddRoundKey operation

**SubBytes Operation;**

This operation substitutes bytes independently by using a nonlinear substitution table called the S-Box table, shown blow:



**ShiftRow Operation:**

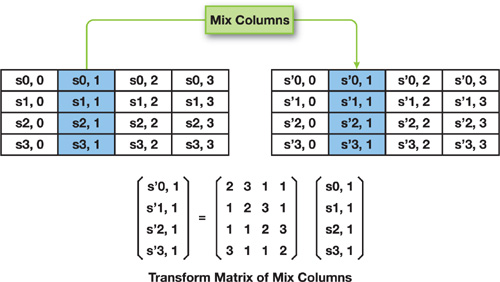
This operation shifts the last three rows of the state cyclically:



**MixColumn Operation:**

The Mixcolumn operation has the purpose of

contending the data of every column. A matrix multiplication is performed on every column to run this operation.



**AddRoundKey Operation:**

It actuates the current round key from the key schedule, where the register serves as the argument.

**Performance of the Cipher:**

As we have known the working of the AES implementation lets measure its performance on GPU based encryption. We have omitted the decryption because it performs the same mechanism as the encryption does.

Our test was performed on the testing machine with the following specification:

* CPU: Pentium 4, 3 GHz, 2 MB Level 2 cache
* Memory: 1 GB
* Video: GeForce 8800 GTS 640 MB
* System: Linux 2.6, Driver 97.46