# AES-GCM

## AES-GCM Authenticated Encrypt/Decrypt Core

The AES-GCM encryption IP core implements Rijndael encoding and decoding in compliance with the NIST Advanced Encryption Standard. It processes 128-bit blocks, and is programmable for 128-, 192-, and 256-bit key lengths.

Four architectural versions are available to suit system requirements. The **Standard** version (AES-GCM-S) is more compact, using a 32-bit datapath and requiring 44/52/60 clock cycles for each data block (128/192/256-bit cipher key, respectively). The **Fast** version (AES-GCM-F) achieves higher throughput using a 128-bit datapath and requiring 11/13/15 clock cycles for each data block depending on key size. For applications where throughput is critical there are two additional versions. The **High Throughput** AES-GCM-X can process 128 bits/cycle and the **Higher Throughput** AES-GCM-X2 can process 256 bits/cycle respectively independent of the key size.

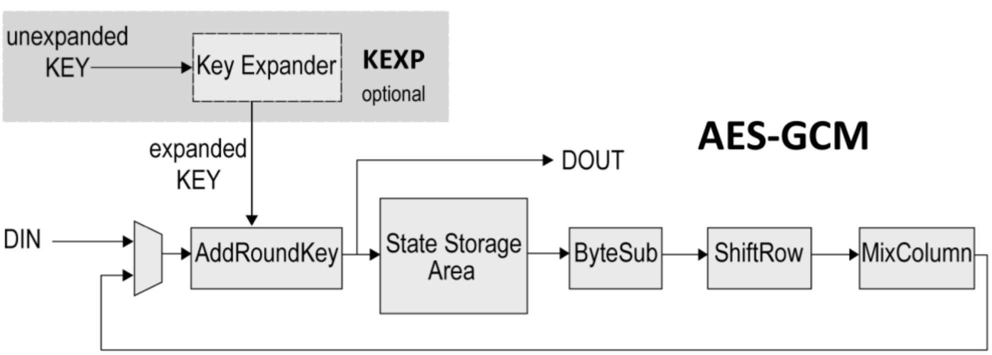
GCM stands for Galois Counter. GCM is a generic authenticate-and-encrypt block cipher mode. A Galois Field (GF) multiplier/accumulator is utilized to generate an authentication tag while CTR (Counter) mode is used to encrypt.

The AES-GCM cores are fully synchronous designs, have been evaluated in a variety of technologies, and are available optimized for ASICs or FPGAs.

### Applications

The AES-GCM can be utilized for a variety of encryption applications including protected network routers, electronic financial transactions, secure wireless communications, secure video surveillance systems, and encrypted data storage.

### Block Diagram




### FEATURES

- Encrypts and decrypts using the AES Rijndael Block Cipher Algorithm
- Implemented according to the National Institute of Standards and Technology (NIST) Special Publication 800-38D
- NIST Certified
- Processes 128-bit data in 32-bit blocks
- Employs user-programmable key size of 128, 192, or 256 bits
- Four architectural versions:
  - AES-GCM-S is more compact: 32-bit data path size
    Processes each 128-bit data block in 44/52/60 clock cycles for 128/192/256-bit cipher keys, respectively
  - AES-GCM-F yields higher transmission rates: 128-bit data path
    Processes each 128-bit block in 11/13/15 clock cycles for 128/192/256-bit cipher keys, respectively
  - Higher throughput versions (AES-GCM-X or AES-GCM-X2) can process 128 bits/cycle or 256 bits/cycle and have a 128-bit datapath size
- 96-bit IV length
- Works with a pre-expended key or can integrate the optional key expansion function
- Simple, fully synchronous, reusable design
- Available as fully functional and synthesizable VHDL or Verilog, or as a netlist for popular programmable devices

Complete deliverables include test benches, C model and test vector generator

### Functional Description

An AES encryption operation transforms a 128-bit block into a block of the same size. The encryption key can be chosen among three different sizes: 128, 192, or 256 bits. The key is expanded during cryptographic operations.

The AES algorithm consists of a series of steps repeated a number of times (rounds). The number of rounds depends on the size of the key and the data block. The intermediate cipher result is known as state.

Initially, the incoming data and the key are added together in the AddRoundKey module. The result is stored in the State Storage area.

| | KSIZE = 00 | KSIZE = 01 | KSIZE = 10 |
|---|---|---|---|
| Rounds | 10 | 12 | 14 |

Number of rounds as a function of key size

The state information is then retrieved and the ByteSub, Shiftrow, MixColumn and AddRoundKey functions are performed on it in the specified order. At the end of each round, the new state is stored in the State Storage area. These operations are repeated according to the number of rounds. The final round is anomalous as the MixColumn step is skipped. The cipher is output after the final round.

### GCM mode

A Galois Field (GF) multiplier/accumulator is utilized to generate an authentication tag while CTR mode is used to encrypt. The counter value is initialized by an IV input from the user.

During encryption or decryption, CTR mode is used to process the incoming data. An authentication tag up to 128 bits long is also produced by hashing with a GF multiply/accumulator additional data as well as the result of the CTR mode.

At the end of decryption, the user should verify that the authentication tag matches the original. If the former is different from the original, the authentication has failed. In this case no other information (i.e. no decrypted data or the value of the authentication tag) should be revealed except the failure itself.

### Key Expansion

The AES algorithm requires an expanded key for encryption or decryption. The KEXP AES key expander core is available as an AES-GCM core option for the standard and fast versions. It is included for the higher throughput versions.

During encryption, the key expander can produce the expanded key on the fly while the AES core is consuming it. For decryption, though, the key must be pre-expanded and stored in an appropriate memory before being used by the AES core. This is because the core uses the expanded key backwards during decryption. In some cases a key expander is not required. This might be the case when the key does not need to be changed (and so it can be stored in its expanded form) or when the key does not change very often (and thus it can be expanded more slowly in software).

## Implementation Results

The AES-GCM can be mapped to any Xilinx FPGA device (provided sufficient silicon resources are available). The following are sample Intel results with all core I/Os assumed to be routed on-chip and throughput for a 128-bit key size.

**AES-GCM Standard Core Xilinx Implementation Results**

| Family | LUTs | BRAMs | Freq. (MHz) | Throughput (Mbps) |
|---|---|---|---|---|
| Virtex-7 (-3) | 907 | 2 | 275 | 800 |
| Kintex-7 (-3) | 910 | 2 | 300 | 873 |
| Kintex UltraScale (-3) | 910 | 2 | 425 | 1,236 |
| Kintex UltraScale+ (-3) | 904 | 2 | 550 | 1,600 |

**AES-GCM Fast Core Xilinx Implementation Results**

| Family | LUTs | BRAMs | Freq. (MHz) | Throughput (Mbps) |
|---|---|---|---|---|
| Virtex-7 (-3) | 1,846 | 8 | 250 | 2.909 |
| Kintex-7 (-3) | 1,846 | 8 | 250 | 2.909 |
| Kintex UltraScale (-3) | 1,808 | 8 | 375 | 4,364 |
| Kintex UltraScale+ (-3) | 1,905 | 8 | 475 | 5,527 |

**AES-GCM High Throughput (-X) Implementation Results**

| Family | LUTs | BRAMs | Freq. (MHz) | Throughput (Gbps) |
|---|---|---|---|---|
| Virtex-7 (-3) | 9,942 | 108 | 250 | 32.0 |
| Kintex-7 (-3) | 9,881 | 108 | 250 | 32.0 |
| Kintex UltraScale (-3) | 11,485 | 108 | 325 | 41.6 |
| Kintex UltraScale+ (-3) | 9,409 | 108 | 350 | 44.8 |

**AES-GCM Higher Throughput (-X2) Impl. Results**

| Family | LUTs | BRAMs | Freq. (MHz) | Throughput (Gbps) |
|---|---|---|---|---|
| Virtex-7 (-3) | 23,064 | 216 | 200 | 51.2 |
| Kintex-7 (-3) | 23,897 | 216 | 200 | 51.2 |
| Kintex UltraScale (-3) | 24,949 | 216 | 250 | 64.0 |
| Kintex UltraScale+ (-3) | 24,809 | 216 | 300 | 76.8 |

## Related Products

AES in CBC, CCM, CFB, CTR, ECB, LRW, OFB and XTS modes are also available as stand-alone cores.

AES-P: run-time programmable AES core supporting ECB, CBC, CFB, OFB and CTR modes.

## Export Permits

This core implements encryption functions and as such it is subject to export control regulations. Export to your country may or may not require a special export license. Please contact CAST to determine what applies in your specific case.

## Support

The core as delivered is warranted against defects for ninety days from purchase. Thirty days of phone and email technical support are included, starting with the first interaction. Additional maintenance and support options are available.

## Verification

The core has been verified through extensive synthesis, place and route and simulation runs. It has also been embedded in several products, and is proven in FPGA technologies.

## Deliverables

The core is available in ASIC (RTL) or FPGA (netlist) formats, and includes everything required for successful implementation. The Xilinx version includes:

- Targeted FPGA netlist; Sophisticated HDL Testbench (self-checking); C Model & test vector generator
- Simulation script, vectors & expected results; Synthesis script; User documentation