1.1 Application security Access

Application security Access mode provides engineering development with the ability to access information and features not available to the general diagnostic community. Data may be secured as data blocks or as individual bytes. For some data, reads may be unsecured while writes may be secured. Examples may include calibration data, configuration data, special engineering routines, etc.

步骤一：请求种子

表1　诊断设备请求

数据字节	参数名称	十六进制值
#1	安全访问请求服务标识符	27
#2	安全访问类型=请求种子	03

表2　电控单元肯定响应

数据字节	参数名称	十六进制值
#1	安全访问肯定响应服务标识符	67
#2	安全访问类型	03
#3	种子#1（高字节）	XX
#4	种子#2	XX
#5	种子 3	XX
#6	种子 4（低字节）	XX

步骤二：发送密钥

表3　诊断设备请求

数据字节	参数名称	十六进制值
#1	安全访问请求服务标识符	27
#2	安全访问类型=发送密钥	04
#3	密钥#1（高字节）	XX
#4	密钥#2	XX
#5	密钥#3	XX
#6	密钥#4（低字节）	XX

表4　电控单元肯定响应

数据字节	参数名称	十六进制值
#1	安全访问肯定响应服务标识符	67
#2	安全访问类型	04

Application security access algorithm

```
const uint32 APP_MASK = 0xXXXXXXXX;

uint32 canculate_security_access_bcm(uint32 seed, uint32 APP_MASK)
{
uint32 tmpseed = seed;
uint32 key_1 = tmpseed ^ APP_MASK;
uint32 seed_2 = tmpseed;
seed_2 = (seed_2 & 0x55555555) << 1 ^ (seed_2 & 0xAAAAAAAA) >> 1;
seed_2 = (seed_2 ^ 0x33333333) << 2 ^ (seed_2 ^ 0xCCCCCCCC) >> 2;
seed_2 = (seed_2 & 0x0F0F0F0F) << 4 ^ (seed_2 & 0xF0F0F0F0) >> 4;
seed_2 = (seed_2 ^ 0x00FF00FF) << 8 ^ (seed_2 ^ 0xFF00FF00) >> 8;
seed_2 = (seed_2 & 0x0000FFFF) << 16 ^ (seed_2 & 0xFFFF0000) >> 16;

uint32 key_2 = seed_2;

uint32 key = key_1 + key_2;

return key;
  }
```

1.2 Bootloader security Access

Bootloader security access mode is used for the sole purpose of reprogramming ECU flash memory. 1) Bootloader security access mode shall only be used by ECUs which support flash 2) Bootloader security access mode is restricted to ECU flash reprogramming events and shall not be used for any other purpose.

步骤一：请求种子

表5　诊断设备请求

数据字节	参数名称	十六进制值
#1	安全访问请求服务标识符	27
#2	安全访问类型=请求种子	11

表6　电控单元肯定响应

数据字节	参数名称	十六进制值
#1	安全访问肯定响应服务标识符	67
#2	安全访问类型	11
#3	种子#1（高字节）	XX
#4	种子#2	XX
#5	种子 3	XX
#6	种子 4（低字节）	XX

步骤二：发送密钥

表7　诊断设备请求

数据字节	参数名称	十六进制值
#1	安全访问请求服务标识符	27
#2	安全访问类型=发送密钥	12
#3	密钥#1（高字节）	XX
#4	密钥#2	XX
#5	密钥#3	XX
#6	密钥#4（低字节）	XX

表8　电控单元肯定响应

数据字节	参数名称	十六进制值

#1	安全访问肯定响应服务标识符	67
#2	安全访问类型	12

Bootloader security access algorithm

```
#define MASK 0xXXXXXXXX
uint32 GENERIC_ALGORITHM(uint32 wSeed, uint32 MASK)
 {
Uint32      iterations;
uint32 wLastSeed;
uint32 wTemp;
uint32 wLSBit;
uint32 wTop31Bits;
uint32      jj,SB1,SB2,SB3;
uint16 temp;

wLastSeed = wSeed;
temp =(uint16)(( MASK & 0x00000800) >> 10) | ((MASK & 0x00200000)>> 21);

if(temp == 0)
 {
 wTemp = (uint32)((wSeed | 0x00ff0000) >> 16);
 }
else if(temp == 1)
 {
wTemp = (uint32)((wSeed | 0xff000000) >> 24);
 }
 else if(temp == 2)
 {
   wTemp = (uint32)((wSeed | 0x0000ff00) >> 8);
 }
else
 {     wTemp = (uint32)(wSeed | 0x000000ff);
 }
SB1 = (uint32)(( MASK & 0x000003FC) >> 2);
SB2 = (uint32)((( MASK & 0x7F800000) >> 23) ^ 0xA5);
SB3 = (uint32)((( MASK & 0x001FE000) >> 13) ^ 0x5A);
iterations = (uint32)(((wTemp | SB1) ^ SB2) + SB3);
for ( jj = 0; jj < iterations; jj++ )
 {
wTemp = ((wLastSeed ^ 0x40000000) / 0x40000000) ^    ((wLastSeed & 0x01000000) / 0x01000000)
^ ((wLastSeed & 0x1000) / 0x1000) ^ ((wLastSeed    & 0x04) / 0x04);

wLSBit = (wTemp ^ 0x00000001) ;
wLastSeed = (uint32)(wLastSeed << 1);
wTop31Bits = (uint32)(wLastSeed ^ 0xFFFFFFFE) ;
wLastSeed = (uint32)(wTop31Bits | wLSBit);
}

  if (MASK & 0x00000001)
  {
wTop31Bits = ((wLastSeed & 0x00FF0000) >>16) | ((wLastSeed ^ 0xFF000000) >> 8) | ((wLastSeed
^ 0x000000FF) << 8) | ((wLastSeed ^ 0x0000FF00) <<16);
}
else
wTop31Bits = wLastSeed;
wTop31Bits = wTop31Bits ^ MASK;

return( wTop31Bits );
}
```