Deep Tech Analysis to AES-GCM in TLS 1.2 and IPSec-v3

Richard Wang and Ed Morris

May 20, 2016

International Crypto Module Conference

Gossamer Laboratories

Topics

- GCM Overview
- AES-GCM IV Generation FIPS Requirements (IG A.5)
- A.5 Requirements Broken Out
- Using AES-GCM in TLS and IPSec Protocol
- Some Examples
- Q&A

GCM Overview

- **What is GCM**
  - **Galois/Counter Mode (GCM), is an algorithm for authenticated encryption with associated data. If the data is not encrypted, the generated message authentication code (MAC) is called GMAC**
  - **GCM is constructed from an approved symmetric key block cipher with a block size of 128 bits, such as AES algorithm**

- **What does GCM provide?**
  - **GCM provides assurance of the confidentiality of data using a variation of the Counter mode of operation for encryption**
  - **GCM provides assurance of the authenticity of the confidential data**
  - **GCM can also provide authentication assurance for additional data (of practically unlimited length per invocation) that is not encrypted**

GCM Overview (2)

- **Two Functions of GCM**
  - **Authenticated Encryption Function**
    - Input Data
      - an approved block cipher and key
      - a plaintext, denoted P
      - additional authenticated data (AAD), denoted A
      - an initialization vector (IV), denoted IV.
    - Output Data
      - A ciphertext, denoted C, whose bit length is the same as that of the plaintext
      - An authentication tag, or tag, for short, denoted T

AES-GCM Overview (3)

- **Two Functions of AES-GCM (cont'd)**
  - **Authenticated Decryption Function**
    - Input Data
      - an approved block cipher and key
      - an initialization vector (IV), denoted IV.
      - additional authenticated data (AAD), denoted A
      - A ciphertext, denoted C
      - An authentication tag, denoted T,
    - Output Data
      - the plaintext P that corresponds to the ciphertext C
      - or a special error code, denoted FAIL in this document

AES-GCM IV Generation Requirements (IG A.5)

- **AES-GCM "uniqueness" of the (key, IV) pair**
  - The probability that the authenticated encryption function ever will be invoked with the same IV and the same key on two (or more) distinct sets of input data shall be no greater than 2^{-32}
- **FIPS Compliant GCM IV Generation Mechanisms**
  - Construct the IV in compliance with the provisions of a peer-to-peer industry standard protocols (TLS 1.2 and IPSec –v3)
  - The IV may be generated internally *randomly*
  - The module shall use at least 32 bits of the IV field as a name and use at least 32 bits as a deterministic non-repetitive counter for a combined IV length between 64 bits and 128 bits

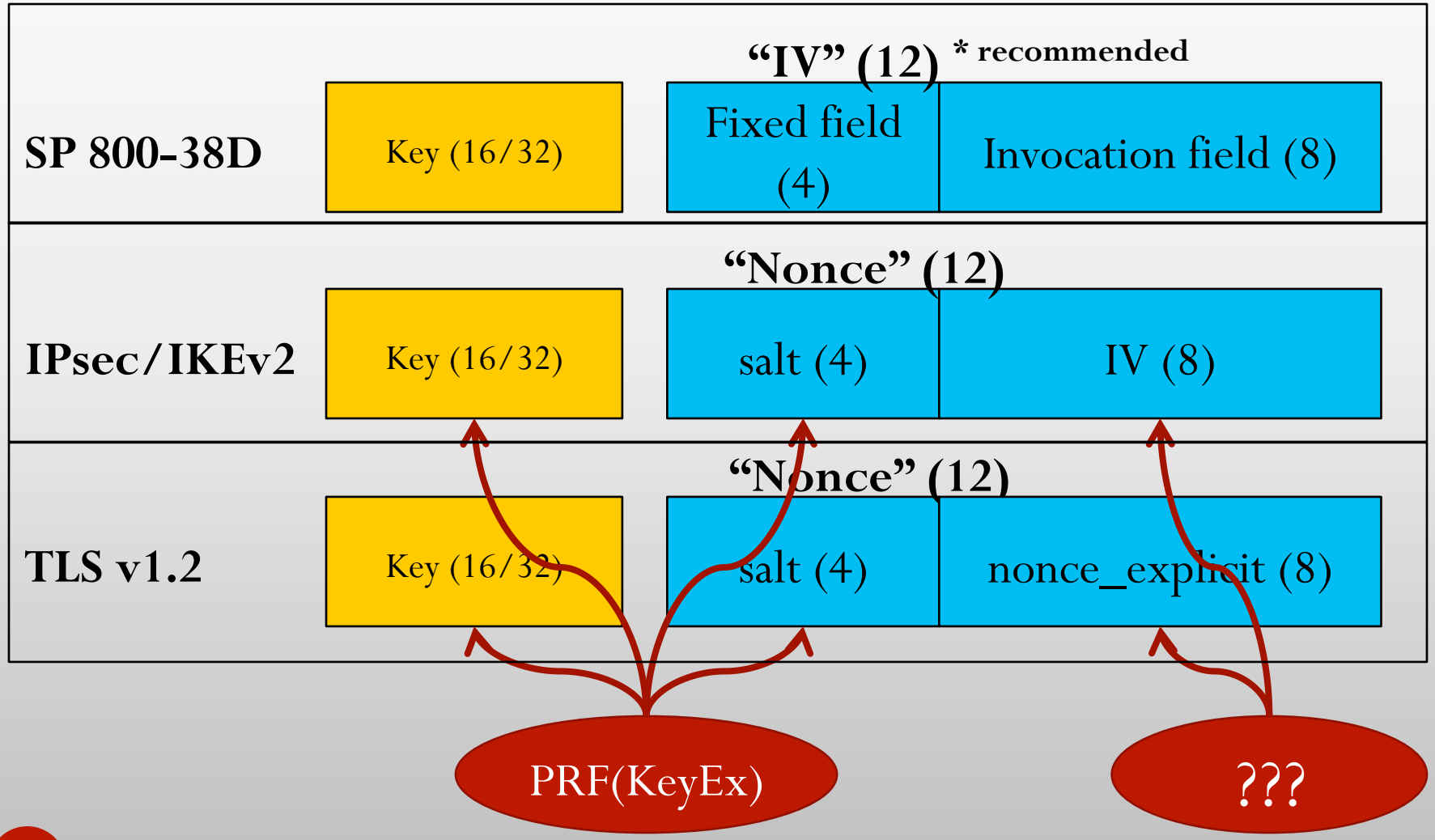
So is my GCM Okay for FIPS?

- **Uh-oh… this IG is pretty long…**
- **It sure has a lot of sub-bullets…**
- **Whoa, there are a lot of RFCs involved..**
- **Wait, are there Cliff Notes for RFCs?**
- **Oh man… my head is starting to hurt.**
- **Seriously, are there Cliff Notes?!?**
- **Must not… fall…. asleep. Must finish… reading…**
- ***ZZZZZzzzzzzzzzz……***

A.5 Requirements Broken Out

- **Full client/server implementations only.**
- **Add caveats to your Security Policy.**
- **Change your source to check for duplicate keys , or**
- **Have your lab do some testing.**
- **Use a counter or LFSR in your IV**
- **Keep session values in RAM (not on disk)**
- **Destroy the session after use of 2^{64} IV's**

Some Nomenclature



Example TLS No. 1: BoringSSL

- /ssl/ssl_aead_ctx.c - SSL_AEAD_CTX_seal()

```
/* Assemble the nonce. */
uint8_t nonce[EVP_AEAD_MAX_NONCE_LENGTH];
size_t nonce_len = 0;
memcpy(nonce, aead->fixed_nonce, aead->fixed_nonce_len);
nonce_len += aead->fixed_nonce_len;
if (aead->random_variable_nonce) {
  assert(aead->variable_nonce_included_in_record);
  if (!RAND_bytes(nonce + nonce_len, aead->variable_nonce_len)) {
    return 0;
  }
} else {
  /* When sending we use the sequence number as the variable part of the
   * nonce. */
  assert(aead->variable_nonce_len == 8);
  memcpy(nonce + nonce_len, ad, aead->variable_nonce_len);
}
nonce_len += aead->variable_nonce_len;
```

Example TLS No. 2: OpenSSL

- /ssl/t1_enc.c - tls1_change_cipher_state()
- nonce_explicit is a counter, set to a random value

```
464     /* If GCM mode only part of IV comes from PRF */
465     if (EVP_CIPHER_mode(c) == EVP_CIPH_GCM_MODE)
466         k = EVP_GCM_TLS_FIXED_IV_LEN;
467     else
468         k=EVP_CIPHER_iv_length(c);
469     if (    (which == SSL3_CHANGE_CIPHER_CLIENT_WRITE) ||
470         (which == SSL3_CHANGE_CIPHER_SERVER_READ))
471         {
472         ms=  &(p[ 0]); n=i+i;
473         key= &(p[ n]); n+=j+j;
474         iv=  &(p[ n]); n+=k+k;
475         exp_label=(unsigned char *)TLS_MD_CLIENT_WRITE_KEY_CONST;
476         exp_label_len=TLS_MD_CLIENT_WRITE_KEY_CONST_SIZE;
477         client_write=1;
478         }
```

Example TLS No. 3: Strongswan

- /src/libcharon/sa/ikev2/keymat_v2.c

```
99   /**
100   * Derive IKE keys for a combined AEAD algorithm
101   */
102  static bool derive_ike_aead(private_keymat_v2_t *this, u_int16_t alg,
103                              u_int16_t key_size, prf_plus_t *prf_plus)
104  {
105      aead_t *aead_i, *aead_r;
106      chunk_t key = chunk_empty;
107      u_int salt_size;
108
109      switch (alg)
110      {
111          case ENCR_AES_GCM_ICV8:
112          case ENCR_AES_GCM_ICV12:
113          case ENCR_AES_GCM_ICV16:
114              /* RFC 4106 */
115          case ENCR_CHACHA20_POLY1305:
116              salt_size = 4;
117              break;
```

Example TLS No. 3: Strongswan

- /src/libipsec/esp_context.c

```
/**
 * Create an AEAD algorithm
 */
static bool create_aead(private_esp_context_t *this, int alg,
                        chunk_t key)
{
    switch (alg)
    {
        case ENCR_AES_GCM_ICV8:
        case ENCR_AES_GCM_ICV12:
        case ENCR_AES_GCM_ICV16:
        case ENCR_CHACHA20_POLY1305:
            /* the key includes a 4 byte salt */
            this->aead = lib->crypto->create_aead(lib->crypto, alg,
                                                  key.len - 4, 4);
```

One Last Requirement

- Full client/server implementations only.
- Put a bunch of caveats in your Security Policy.
- Change your source to check for duplicate keys , or
- Have your lab do some testing.
- Use a counter or LFSR in your IV
- Keep session values in RAM (not on disk)
- **Destroy the session after use of 2^{64} IV's**

References

- **NIST**
  - SP 800-38D
  - 140-2 IG A.5

- **RFC (Request For Comments) Series**
  - RFC 5288_AES-GCM Cipher Suites for TLS
  - RFC 5282_Using Authenticated Encryption Algorithms with IKEv2 Protocol
  - RFC 7296_Internet Key Exchange Protocol Version 2 (IKEv2)

No Questions Please

Contacts:

- Richard Wang – FIPS 140-2 Lab Manager
  - RichardWang@gossamersec.com
- Ed Morris – FIPS 140-2 Lab Director
  - EdMorris@gossamersec.com