Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 18, 2014

D. McGrew
Cisco Systems, Inc.
October 15, 2013

Generation of Deterministic Initialization Vectors (IVs) and Nonces
draft-mcgrew-iv-gen-03.txt

Abstract

Many cryptographic algorithms use deterministic IVs, including CTR, GCM, CCM, GMAC. This type of IV is also called a (deterministic) nonce. Deterministic IVs must be distinct, for each fixed key, to guarantee the security of the algorithm. This note describes best practices for the generation of such IVs, and summarizes how they are generated and used in different protocols. Some problem areas are highlighted, and test considerations are outlined. This note will be useful to implementers of algorithms using deterministic IVs, and to protocol or system designers using them.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction 3
 1.1. Conventions Used In This Document 3
2. Deterministic IVs in Algorithms 4
3. Deterministic IVs in Protocols 5
4. Deterministic IVs in Standards 7
 4.1. Recommended IV/Nonce Format 7
 4.2. Partially Implicit IV/Nonce Format 8
 4.3. Alternative IV/Nonce Format 9
 4.4. Unpredictable IV/Nonce Format 10
 4.5. ESP 11
 4.6. IKE 12
 4.7. TLS 12
 4.8. SSH 12
 4.9. SRTP 12
 4.10. Summary 12
5. Implementation 14
 5.1. IV Verification 16
6. Testing 17
 6.1. Internal IV Generator 17
7. Issues 18
 7.1. Choice of Fixed-Distinct Field 18
 7.2. Size of the Fixed-Distinct Field 19
 7.3. Security 19
 7.4. Bandwidth Use 20
8. Security Considerations 21
9. Acknowledgments 22
10. References 23
 10.1. Normative References 23
 10.2. Informative References 23
Author's Address 26

1. Introduction

This note describes deterministic IVs and nonces and how they are used in cryptographic algorithms (Section 2), then describes their use in protocols (Section 3), and then their use in standards (Section 4). Considerations for implementation (Section 5) and testing (Section 6) are presented. Issues and potential problems are discussed (Section 7). The focus is on network security protocols, rather than on the security of data at rest, though many of the same considerations apply in both areas.

1.1. Conventions Used In This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Deterministic IVs in Algorithms

Many cryptographic algorithms use Initialization Vectors, or IVs. An IV is provided to an algorithm along with a message to be processed; the IV initializes the algorithm to process the message. Typically, there will be many IVs that are used with a single key. Some algorithms, such as the Cipher Block Chaining (CBC) encryption mode, require that the IVs that it uses are completely unpredictable. Such IVs are typically called random IVs, and they must be generated by a cryptographically strong random or pseudorandom process [RFC4086].

Another type of IVs are deterministic IVs. These IVs are generated by a deterministic process. The classic example of an algorithm that uses a deterministic IV is counter (CTR) mode encryption [CTR]. An algorithm that uses deterministic IVs requires that each IV provided as input to the algorithm be distinct, for a fixed key.

A deterministic IV is sometimes called a nonce, or a deterministic nonce. In cryptography, a nonce is a value that is used only once. Many cryptographic protocols include a nonce in a message to enable its receiver to recognize whether or not the message has been previously received and processed. From the point of view of a cryptographic algorithm that uses deterministic IVs, calling the IV a nonce emphasizes the role of the IV in the overall system or protocol. Calling that value a deterministic IV emphasizes its role in initializing the algorithm to process a new message. Nonetheless, these are just different monikers for the same thing.

Authenticated Encryption is a symmetric encryption method that provides for the authenticity and integrity of the data that it protects, as well as its confidentiality [BN00] [R02]. An authenticated encryption method that uses deterministic IVs will need to make sure that the IVs used for encryption are distinct. However, when performing the decryption operation, there is no need to ensure that the IVs are distinct; the authenticated decryption operation does not impose that requirement. The Authenticated Encryption methods used in standards include Galois Counter Mode [GCM] and Counter and CBC MAC mode [CCM].

Some Message Authentication Code (MACs) use deterministic IVs, including GMAC [GCM] and UMAC [RFC4118]. The considerations for Authenticated Encryption also apply to these MAC algorithms: the IVs used in the generation of an authentication tag must be distinct, but there is no need to verify the distinctness of an IV prior to inputting that IV to a tag verification algorithm.

3. Deterministic IVs in Protocols

The simplest way to implement a deterministic IV or nonce is to use a counter: initialize an integer variable to zero, then each time that an IV is needed, output the integer value, then store the incremented value after checking to make sure that no integer overflow occurred, so that no counter value is used twice. The simplicity of this method has made it popular in practice, and recommended by standards.

The straightforward method of using a counter is not sufficient when there are multiple encryption engines that are using the same encryption key. This can be the case when encryption is distributed across multiple processors, or across multiple software threads, processes, or virtual machines. It can also happen in cases where a protocol allows group keys. In these cases, some mechanism is needed that ensures that IVs are distinct across all encryption engines that use the same key. This is easily accomplished by including a fixed field in the IV that is distinct for each distinct encrypter. (This is detailed in Section 4.1.)

When a deterministic IV is used to encrypt and/or authenticate a message, the receiver(s) of that message needs to know that IV in order to decrypt it and/or verify its authenticity. A deterministic IV can be sent along with a message, which makes it plain to the receiver(s), or it can be left out of a message if the receiver(s) have enough information to reconstruct it. Leaving the IV out of the message reduces the amount of data that must be communicated, which is advantageous. On the other hand, if the IV is included in the message, the receiver(s) need not be aware of the method by which the sender has chosen the IVs.

In practice, some protocols have split the difference between the implicit method (in which the IV is absent and a receiver infers its value) and the explicit method (in which the entire IV is included with the message). The IV is constructed out of two fields: an explicit field, which is conveyed along with the message, and an implicit field, which is coordinated between the encrypter and the decrypter using an "out of band" method. (This is detailed in Section 4.2.) In most cases, the key management protocol that establishes the encryption key can also establish the implicit field.

In a block cipher mode of operation that use deterministic IVs, the inputs to each of the block cipher invocations during the encryption process are determined by the IV provided to that process. It is desirable to make the inputs to the block cipher unpredictable to an attacker, to the extent that is possible, to make cryptanalytic attacks more difficult and costly to attackers. This is true for several types of attacks, including time-memory tradeoff attacks and

key collision attacks [MF00], which are generic attacks that can be applied to any cipher, and cipher-specific attacks such as integral cryptanalysis [KW02]. (It is worth noting that counter mode gives an attacker exactly what they want for integral cryptanalysis: a complete set of block cipher inputs that differ only in some bit positions.) The cost of these attacks can be significantly increased by making the deterministic IV unpredictable to potential attackers. This security benefit is one motivation for why the implicit field of the deterministic IV is kept secret in some protocols.

It is not hard to adapt the simple methods for constructing deterministic IVs so that they produce IVs that are unpredictable. An easy way to do that is to have a secret value that is bitwise exclusive-ored into the IV after all of the other processing is done. (This is detailed in Section 4.4.) This secret value must be known to all encrypters and decrypters, and be established via some "out of band" mechanism. In practice, it is typically established by the key management system.

4. Deterministic IVs in Standards

Many different protocols use deterministic IVs, including ESP [RFC4106], TLS [RFC5288], SSH [RFC5647], and SRTP [RFC3711]. The way that these protocols define their IVs is outlined in this section and is summarized in Table 1.

4.1. Recommended IV/Nonce Format

RFC 5116 defines the interface for Authenticated Encryption, which is the most common use of deterministic IVs at present. That RFC recommends an IV format that is used by ESP, IKE, TLS, and SSH. The recommended format has a total length of 12 octets, and consists of a Fixed Field and a Counter field, and is structured as in Figure 1. (See Section 3.2 of [RFC5116] for the precise normative description.)

```
   +-------------------------------+------------------------+
   |             Fixed             |         Counter        |
   +-------------------------------+------------------------+
```

Figure 1: Recommended IV/Nonce format.

```
                  Fixed      Counter
                 <------><-------------->
         1st     5DAD87F80000000000000001
         2nd     5DAD87F80000000000000002
         3rd     5DAD87F80000000000000003
         4th     5DAD87F80000000000000004
         5th     5DAD87F80000000000000005
         ...                ...
```

Figure 2: An example output of recommended IV/nonce format, showing successive IVs where the Fixed field is 5DAD87F8.

The Fixed field remains constant for all nonces that are generated for a given encryption device. If different devices are performing encryption with a single key, then each distinct device MUST use a distinct value for the Fixed field, to ensure the uniqueness of the nonces that it generates.

This format is suggested, but not required, by [CTR].

With this format, the Counter fields of successive nonces form a monotonically increasing sequence, when those fields are regarded as unsigned integers in network byte order.

4.2. Partially Implicit IV/Nonce Format

The case in which the recommended format is used with Partially Implicit Nonces has further details. In that case, the IV is structured as in Figure 3.

```
   +--------------+----------------+------------------------+
   | Fixed-Common | Fixed-Distinct |         Counter        |
   +--------------+----------------+------------------------+
    <- implicit -> <--------------- explicit -------------->
```

Figure 3: Partially implicit IV/Nonce format

```
                  Fixed  Fixed
                  Common Distinct  Counter
                  <------><--><---------->
        1st       5DAD87F81E0E000000000001
        2nd       5DAD87F81E0E000000000002
        3rd       5DAD87F81E0E000000000003
        4th       5DAD87F81E0E000000000004
        5th       5DAD87F81E0E000000000005
        ...                 ...
```

Figure 4: An example output of Partially Implicit IV/Nonce format, showing successive IVs where the Fixed-Common field is 5DAD87F8 and the Fixed-Distinct field is 1E0E.

The portion of the IV that is stored or sent with the ciphertext is the explicit part. The portion of the IV that is not sent with the ciphertext is the implicit part.

The Fixed field is divided into two sub-fields: a Fixed-Common field and a Fixed-Distinct field.

If different devices are performing encryption with a single key, then each distinct device MUST use a distinct Fixed-Distinct field. The Fixed-Common field is common to all IVs. The Fixed-Distinct field and the Counter field MUST be in the explicit part of the IV. The Fixed-Common field MAY be in the implicit part of the IV.

ESP, IKE, TLS, and SSH conform to the alternative IV/nonce format, though in practice the partially implicit format is often used. Those standards do not require that the "Changing" field actually be a counter (instead, "anything that guarantees uniqueness can be used"), but in practice a counter is convenient.

The partially implicit format can save on bandwidth or data storage

requirements, because it avoids sending or storing the implicit part of the IV. However, it limits the number of IVs that can be generated, because the implicit part is fixed, and it adds complexity to the system, by making the system coordinate the implicit part through out-of-band means. Thus, new protocol and system designs SHOULD NOT use the partially implicit format unless a review of all of the issues shows that the bandwidth or storage savings are worth the complexity. (An alternative strategy for bandwidth savings is discussed in Section 7.4.)

4.3. Alternative IV/Nonce Format

In some cases, it may be desirable to avoid the use of a network byte order monotonically increasing counter. This would be especially true in a protocol that has the security goal of obscuring the relationship between messages, so that an attacker cannot infer that particular messages belong to the same flow, and cannot infer the order of messages within a flow. In other cases, there may be a data element available that meets the requirement of being distinct for each invocation of the encryption operation, but is not monotonically increasing in network byte order. In such situations, it makes sense to use an alternative nonce format that replaces the Counter field, as it is used above, with a field that is distinct for each IV/nonce that is generated, but which is not a counter. We call this field the Changing field. The alternative format is shown in Figure 5, and an example output is shown in Figure 6.

IVs/Nonces that are in the partially implicit format also happen to conform to the alternative format as well.

```
+--------------+----------------+-----------------------+
| Fixed-Common | Fixed-Distinct |        Changing       |
+--------------+----------------+-----------------------+
 <- implicit -> <--------------- explicit -------------->
```

Figure 5: Alternative IV/Nonce format

```
                       Fixed
                       Common     Changing
                      <------><-------------->
            1st       88be20d4600ced63f924ff5b
            2nd       88be20d458a0169be27d661f
            3rd       88be20d43ece0f6a4061eca8
            4th       88be20d46016f0a41c1cbc27
            5th       88be20d4ba7f697eb00aee67
            ...                 ...
```

Figure 6: An example output of Alternative IV/Nonce format, showing successive IVs where the Fixed-Common field is 5DAD87F8 and the Fixed-Distinct field is zero length (or equivalently, is not present).

4.4. Unpredictable IV/Nonce Format

This method is shown in Figure 7, in which the symbol (+) denotes the bitwise exclusive-or operation. (Here the Fixed field consists of the Fixed-Common field followed by the Fixed-Distinct field.) This format uses a Randomizer, which is an octet string that is combined with the other fields to make the IVs unpredictable. The length of the Randomizer must be no greater than the sum of the lengths of the Fixed and Counter fields.

The next IV in sequence is computed as follows. The Fixed field and the Counter field are concatenated. If the length of the Randomizer is less than the combined length of the Fixed and Counter fields, then the Randomizer is padded on the right with enough zeros so that the padded value has a length that exactly matches that of the Fixed and Counter fields together. The concatenated Fixed and Counter field is bitwise exclusive-ored with the padded Randomizer, and the resulting value is the IV. The Counter is incremented, treating it as an unsigned integer with the most significant byte on the left, and the stored Counter field is set to the incremented value. Then the IV is returned. This is the method used by SRTP [RFC3711], wherein the Randomizer field is called "Salt". (We use the term Randomizer instead of Salt, because the latter term is used with slightly different meanings in some other specifications, such as [RFC4309].)

```
                 +-----------------+-----------------+
                 |     Fixed       |      Counter    |---+
                 +-----------------+-----------------+   |
                                                         |
                 +-----------------------------------+   v
                 |             Randomizer            |->(+)
                 +-----------------------------------+   |
                                                         |
                 +-----------------------------------+   |
                 |       Initialization Vector       |<--+
                 +-----------------------------------+
```

Figure 7: Unpredictable IV/Nonce Format.

```
                Fixed  Fixed
                Common Distinct   Counter            IV
                 <--><------><----------->  <----------------------->
       1st       000097B4AE8F000000000001  0C81C77A5DDB678EE16FA2D0
       2nd       000097B4AE8F000000000002  0C81C77A5DDB678EE16FA2D3
       3rd       000097B4AE8F000000000003  0C81C77A5DDB678EE16FA2D2
       4th       000097B4AE8F000000000004  0C81C77A5DDB678EE16FA2D5
       5th       000097B4AE8F000000000005  0C81C77A5DDB678EE16FA2D4
       ...                     ...
```

Figure 8: An example output of the Unpredictable IV/nonce format, showing successive IVs where the Fixed-Distinct field has the value 97B4AE8F and the Salt has value 0C8150CEF354678EE16FA2D1.

4.5. ESP

In the IP Encapsulating Security Payload (ESP) [RFC3686][RFC4106][RFC4309] the implicit and explicit parts are four and eight bytes long, respectively. The exception is [RFC4309], for which the implicit part is three bytes in length. The Fixed-Common field is four bytes, and its value is set by the Internet Key Exchange (IKE). (This field is named inconsistently, being called Nonce in [RFC3686], and Salt in [RFC4106] and [RFC4309].) When ESP is used with IKE, there is exactly one entity performing encryption, and the Fixed-Distinct part is usually not present (or equivalently, is has a length of zero bytes). When ESP is used with a group key management protocol such as GDOI, the Fixed-Distinct field may be two or four bytes in length, and the value of the Fixed-Distinct field to be used by an encrypter is established by the group key management protocol [RFC6054]. The case in which IKE is used with ESP and there are multiple encryption engines is not specifically addressed by the standards, but it can be handled by the use of a nonzero Fixed-

Distinct field.

4.6. IKE

The Internet Key Exchange (IKE) [RFC5282] uses the recommended IV/nonce format. The Fixed-Common field is four bytes in length, and its value is set from the IKE Keying Material. The Fixed-Distinct part is usually zero bytes, but it may be any number of bytes if there are multiple encrypters in use.

4.7. TLS

In Transport Layer Security (TLS) [RFC5288], the Fixed-Common field is four bytes in length, and the Fixed-Distinct part is usually zero bytes, but it may be any number of bytes when there are multiple encrypters in use. Section 6.2 of [RFC5288] gives an example of TLS deterministic IV formation.

4.8. SSH

In the Secure Shell (SSH) protocol [RFC5647] the Fixed-Common field is not present, the Fixed-Distinct field is four bytes long, and the Counter field is eight bytes in length. The implicit part is not present, and the explicit part contains the entire 12 byte IV.

4.9. SRTP

The Secure Real-time Transport Protocol (SRTP) [RFC3711] and draft-ietf-avt-srtp-aes-gcm-01 use the unpredictable format, which is a bit more complex than RFC 5116. It is essentially RFC 5116 format with the additional step of performing a bitwise exclusive-or operation with a Randomizer value. (This step provides additional strength against cryptographic attacks that rely on predicting all or most of the IV.) draft-ietf-avt-srtp-aes-gcm-01 uses a 12-byte IV, though RFC 3711 uses a 14-byte IV.

4.10. Summary

The following table gives a synopsis of how standard protocols use deterministic IVs.

Protocol	IV (bytes)	Fixed-Common (bytes)	Fixed-Distinct (bytes)	Counter (bytes)
ESP	12	4	0,1,2,[4]	8,7,6,[4]
		Not on wire	On wire	On wire
		Set by IKE		
ESP	11	3	0,1,2,[4]	8,7,6,[4]
[RFC4309]		Not on wire	On wire	On wire
		Set by IKE		
IKE	12	4	Unspecified	Unspecified
		Not on wire	On wire	On wire
		Set from KM		
TLS	12	4	0-8	8-0
		Not on wire	On wire	On wire
		Set by TLS		
SSH	12	0	4	8
			On wire	On wire
			Unspecified	
SRTP-CTR	14	4	4	6
		Not on wire	Not on wire	Not on wire
		Set by KM		
SRTP-GCM	12	2	4	6
		Not on wire	Not on wire	Not on wire
		Set by KM		

Table 1: Fields in Deterministic IVs, by Protocol.

5. Implementation

A cryptographic implementation typically consists of a self-contained and testable module that implements all of the essential functionality that it needs. This functionality should include the generation of deterministic IVs.

Because of the variety of ways in which IVs are formed in different protocols, implementers may be tempted to put the generation of the IV under the control of the protocol implementation. That is, from the point of view of the application making use of the encryption algorithm, the IV is an input to that algorithm, as shown in Figure 9. Regardless, it is not good for security to have the IV be generated outside the crypto module. It is possible to implement an IV Generator that can be used with all of the protocols outlined above and use it inside of a cryptographic module. In the following we outline how that can be done.

```
        +----------------------+
        |  +--------------+    |      IV        +-------------+
        |  |              |<-------------------|             |
        |  |  Encryption  |    |   Plaintext   |             |
        |  |  Algorithm   |<-------------------| Application |
        |  |              |    |   Ciphertext  |             |
        |  |              |------------------->|             |
        |  +--------------+    |               +-------------+
        |                      |
        | Cryptographic Module |
        +----------------------+
```

Figure 9: Architecture with IV generation outside of the cryptographic module, showing how the IV is entered into the cryptographic module during an encryption operation.

The internal IV generator architecture is illustrated in Figure 10. The cryptographic module contains an IV Generator sub-module that understands the IV formats outlined in Section 3. To initialize the IV generator, the application inputs the parameter values to be used. Once initialized, the IV generator will produce successive IVs on request, and send these values to the algorithm and to the calling application. The encryption algorithm will need the entire IV, but if the partially implicit IV format is in use, only the explicit part of the IV needs to be provided to the application. The IV generator is responsible for ensuring the distinctness of all of the IVs that it generates.

```
           +----------------------+
           |  +--------------+    |
           |  | IV Generator |-----------+
           |  +--------------+    |      | IV (explicit part)
           |         | IV         |      |
           |         v            |      |
           |  +--------------+    |      |        +-------------+
           |  |              |           +------->|             |
           |  |  Encryption  |    |   Plaintext   |             |
           |  |  Algorithm   |<-------------------| Application |
           |  |              |    |   Ciphertext  |             |
           |  |              |------------------->|             |
           |  +--------------+    |               +-------------+
           |                      |
           | Cryptographic Module |
           +----------------------+
```

Figure 10: Architecture with IV generation inside of the cryptographic module, showing how the IV is generated internally during an encryption operation.

More formally, an IV generator supports the operations of Initialize and Output Next IV. The Initialize operation prepares an IV Generator for use with a particular set of parameters. It takes the following inputs:

A nonnegative integer indicating the number of bytes in the IV to be generated. All of the IVs output from the Generator will have the same length.

An octet string indicating the Fixed part of the IV; this value will be used as the initial part in each IV that is generated.

A nonnegative integer indicating the number of bytes in the Fixed part of the IV. This value must be no greater than the number of bytes in the IV.

An octet string indicating the salt value to be exclusive-ored with the other fields of the IV. If no salt is to be used when Generating IVs, then this parameter must not be present.

A nonnegative integer indicating the number of bytes in the salt value. If no salt value is used, this parameter must be zero. If a salt value is used, this parameter must be no greater than the number of bytes in the IV.

The Fixed field consists of the Fixed-Common field, followed by the Fixed-Distinct field. The Fixed field and Salt field are stored when

the IV generator is initialized; at that time, the Counter field is initialized to zero. The length of the Counter field is equal to the length of the IV less the length of the Fixed field. If the Salt field is shorter than the IV, then it is padded on the right with zeroes. If no Salt is to be used, this is conceptually equivalent to having a Salt value that is the all-zero value.

The Output Next IV operation returns the next IV in sequence, or it returns an indication that there are no more IVs that are available. During that operation, the IV is computed as follows. First, the stored Counter value MUST be checked to determine if an IV can be generated; an IV can only be generated if the value of Counter + 1 does not exceed the maximum allowable value of the Counter field. If an IV cannot be generated, then the operation returns an indication that there are no more IVs that are available. Otherwise, the Fixed field and the Counter field are concatenated, then they are bitwise exclusive-ored with the Salt field, and the resulting value is the IV. The Counter is incremented, treating it as an unsigned integer with the most significant byte on the left, and the stored Counter field is set to the incremented value. Then the IV is returned.

The IV generator should also be able to output the length of the explicit field, so that an algorithm can output only the explicit part, when that is appropriate.

5.1. IV Verification

In some protocols, the IV is constructed out of fields in the protocol in such a way that it is difficult to have the IVs generated inside of the cryptographic module, without requiring that module to contain protocol-specific logic. In this case, assurance of the uniqueness of IVs can be provided by having the IVs be generated by the protocol, but checked by the cryptographic module.

This approach is taken by many implementations of Secure RTP [RFC3711]. The IV in that protocol is constructed in a way that incorporates a sender identifier (the SSRC field) and the protocol's sequence number. To check the sequence number for uniqueness, an implementation can make use of the anti-replay checking that the protocol uses to check inbound packet. An encrypter can use this approach as well, to make sure that the sequence number used to construct the IV is unique. (Of course, it is necessary to have an IV construction method such that the uniqueness of the sequence number ensures the uniqueness of the IV.) Since many cryptographic protocols contain a function to perform anti-replay check based on a sequence number, this is a convenient strategy.

6. Testing

The testing of a cryptographic module is an important step in assessing the assurance of that module. The IV Generator defined in Section 5 can be tested by an external system to verify that it is operating correctly.

Any IV format can be tested by verifying that all of the IVs are distinct. There are many ways that this can be done; for instance, the command "sort | uniq -d" on POSIX systems can be used to detect repeated lines in a file.

The recommended format can be tested by verifying that the Counter field consists of monotonically increasing values.

An important aspect of an IV generator is that, when it has an N byte Counter or Changing field, it should not generate more than (256)^N IVs. This property should be tested for small values of N (at least 1, 2, and 3), by calling the Output Next IV operation M times, for some M > (256)^N. Note that some implementations may produce fewer than (256)^N IVs, e.g. due to their handling of the all-zero IV. That would not affect security.

6.1. Internal IV Generator

When a cryptographic module uses an internal IV generator, only the explicit part of the IV needs to be output from the module. It is possible to test this use of the IV generator by interacting with an encryption algorithm that uses it (or an Authenticated Encryption algorithm, or a MAC).

The encryption operation takes as input a plaintext, and returns a ciphertext and the explicit part of the IV. To test that the IV generator is working properly, call the encryption operation repeatedly, each time with the same plaintext value, and verify that 1) all of the ciphertexts returned are distinct, and 2) all of the explicit parts that are returned are distinct. The plaintext must be at least 32 bytes long, in order to avoid false positives.

7. Issues

7.1. Choice of Fixed-Distinct Field

When considering what data should go into a Fixed-Distinct field, it is tempting to use system values such as network addresses because they appear to meet the criteria of uniqueness. However, there are several significant problems with this idea. System values that are taken from outside the cryptographic module may not actually be distinct, especially if an attacker can influence the system. System values can also change over time; even if they are actually distinct, they may not be fixed. Lastly, the cryptographic system should have the freedom to put distinct data into the Fixed-Distinct fields, so that it can accommodate multiple encryption engines when they occur.

Internet Protocol (IP) version four addresses are four bytes in length, and thus can fit into the Fixed-Distinct field of a 12-byte IV. However, an IP address is highly unsuitable for this purpose. Most networked devices use dynamically assigned IP addresses, with address assignment via an automatic configuration protocol such as the Dynamic Host Configuration Protocol (DHCP). The addresses are determined by an external system and are communicated over an insecure protocol; furthermore, a DHCP address is only valid for a particular period of time, and may change after that lease has expired. Even when an automatic configuration protocol is not in use, IP addresses are determined by the networking subsystem, and are not under the control of the cryptographic module. Network Address Translation (NAT, [RFC1361]) is commonly used to modify the IP addresses of packets as they traverse a network boundary, for instance between a private address space [RFC1918] and the Internet. Because of NAT, the IP address associated with a particular device will not be consistent throughout the network. Multiple devices can use the same addresses; this technique is utilized in order to provide redundancy or load sharing (see the Virtual Router Redundancy Protocol [RFC3768] for instance). Lastly, IPv4 is currently being replaced by version six of that protocol. IPv6 addresses are sixteen bytes long; this is too long for inclusion in an IV, and the coexistence of both versions on the Internet is likely to increase the use of NAT for protocol translation [RFC6146]. In summary, IP addresses are neither fixed nor distinct, and should not be used in a Fixed-Distinct field.

Similar considerations hold for link layer addresses, Domain Name System (DNS) names, and TCP, UDP, and SCTP ports.

A much better solution is to have the Fixed-Distinct field be assigned by the security system. For instance, if a cryptographic module has multiple encrypters, it can assign that field

appropriately for each encrypter.

7.2. Size of the Fixed-Distinct Field

Deterministic IVs typically have an explicit part that is eight bytes in length. (This size is natural to use with a block cipher that has a 16 byte block width, because no more than (256)^8 packets can be encrypted under a single key without encountering security degradation due to the birthday paradox.) Because the Fixed-Distinct field must appear in the explicit part, larger Fixed-Distinct fields will reduce the number of IVs that can be generated. This can be problematic, especially for high throughput situations. For instance, the ESP protocol allows for up to 2^64 packets to be encrypted under a single key, so it is desirable to use a Counter field that is close to eight bytes in length; this is why [RFC6054] encourages the use of short values in the Fixed-Distinct field. Table 2 presents the lifetimes of a single key that can encrypt 2^32 packets, i.e. a key being used with a four-byte Counter field. At high data rates, keys must be replaced quickly.

	Best Case	Typical Case	Worst Case
	9000 byte packets	850 byte packets	64 byte packets
1 Gbps	3 days	8.6 hours	66 minutes
10 Gbps	8.6 hours	52 minutes	6.6 minutes
40 Gbps	22 minutes	13 minutes	1.6 minutes
100 Gbps	8.9 minutes	5.2 minutes	0.7 minutes

Table 2: Key Lifetimes with a four-byte Counter field

7.3. Security

As long as each deterministic IV is distinct, for each key, then security is assured. However, when deterministic IVs are not distinct, security suffers.

The number of deterministic IVs is limited, regardless of how those IVs are generated. What does an encrypter do when no more IVs are available? It should retire the key that it is currently using, and establish another one. This is the reason that the IETF Guidelines for Cryptographic Key Management [RFC4107] require that automated key management be used for algorithms with deterministic IVs. For

network security protocols, this has proven to be an effective strategy.

Particular care must be taken in Virtual Machine (VM) environments, because the VM cloning and rollback processes can cause inadvertent re-use of deterministic IVs. This is just one of many security problems that can result from uncritical application of VM mechanisms when cryptography is in use [GR05].

7.4. Bandwidth Use

An implicit or partially implicit IV uses less bandwidth than a full-sized IV. But as noted above, the (partially) implicit IV format reduces the number of IVs that can be generated and adds complexity to the system.

An alternative approach to bandwidth savings in a protocol design is to use a predictable IV format, such as that of Section 4.1, and then apply header compression to the IV. Header compression is often used on bandwidth-constrained links, and it can be applied to encrypted packets [RFC3095]. The format of Section 4.1 can easily be handled by header compression. This approach has several benefits: it makes IV generation simpler, it allows bandwidth savings for environments in which it matters while putting the complexity burden onto the systems that opt to realize those savings, and it increases the number of IVs that can be used. Specifications that use this design alternative SHOULD require the use of the IV format in Section 4.1.

8. Security Considerations

Cryptographic algorithms that rely on deterministic IVs or nonces must ensure the uniqueness of those values. The recommendations in this note aim to help implementers achieve that goal.

Implementations should use the nonce formats described in Section 3. The way in which these formats are used in standards is summarized in Table 1.

Implementations should use the internal IV generator described in Section 5.

Almost all cryptographic systems can implement counter-based deterministic IVs. In many cases, it is straightforward to generate deterministic IVs associated with a short-term key in use by a single encrypter, as in a typical point-to-point protocol. Complications can arise, however, when there are multiple encrypters, or when a key is used for an extended period of time. Cryptographic systems that cannot ensure IV distinctness should not use deterministic IVs, and should instead use a misuse-resistant mode of operation such as the Synthetic Initialization Vector (SIV) Authenticated Encryption mode of operation [RFC5295], or a randomized algorithm such as the CBC mode of operation (though an additional authentication mechanism must be used with that option). If authentication but not encryption is required, then it is possible to use an algorithm that does not require an IV, such as HMAC [RFC2104].

9. Acknowledgments

Thanks to Greg Zaverucha and Peter Gutmann for comments.

10. References

10.1. Normative References

[CCM] Dworkin, M., "NIST Special Publication 800-38C: The CCM Mode for Authentication and Confidentiality", U.S. National Institute of Standards and Technology http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C.pdf.

[GCM] Dworkin, M., "NIST Special Publication 800-38D: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC.", U.S. National Institute of Standards and Technology http://csrc.nist.gov/publications/nistpubs/800-38D/SP800-38D.pdf.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

[RFC4106] Viega, J. and D. McGrew, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)", RFC 4106, June 2005.

[RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, January 2008.

[RFC5288] Salowey, J., Choudhury, A., and D. McGrew, "AES Galois Counter Mode (GCM) Cipher Suites for TLS", RFC 5288, August 2008.

[RFC5647] Igoe, K. and J. Solinas, "AES Galois Counter Mode for the Secure Shell Transport Layer Protocol", RFC 5647, August 2009.

10.2. Informative References

[BN00] Bellare, M. and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm", Proceedings of ASIACRYPT 2000, Springer-Verlag, LNCS 1976, pp. 531-545 http://www-cse.ucsd.edu/users/mihir/papers/oem.html.

[CTR] Dworkin, M., "NIST Special Publication 800-38: Recommendation for Block Cipher Modes of Operation", U.S. National Institute of Standards and Technology http://

csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf.

[GR05] Garfinkel, T. and M. Rosenblum, "When Virtual is Harder than Real: Security Challenges in Virtual Machine Based Computing Environments", Proceedings of the 10th Workshop on Hot Topics in Operating Systems http://www.stanford.edu/~talg/papers/HOTOS05/virtual-harder-hotos05.pdf.

[KW02] Knudsen, L. and D. Wagner, "Integral Cryptanalysis", 9th International Workshop on Fast Software Encryption (FSE '02) http://eprint.iacr.org/2004/193, December 2001.

[MF00] McGrew, D. and S. Fluhrer, "Attacks on Additive Encryption of Redundant Plaintext and Implications on Internet Security", Proceedings of the Seventh Annual Workshop on Selected Areas in Cryptography (SAC 2000) Spinger-Verlag.

[R02] Rogaway, P., "Authenticated encryption with Associated-Data", ACM Conference on Computer and Communication Security (CCS'02), pp. 98-107, ACM Press, 2002. http://www.cs.ucdavis.edu/~rogaway/papers/ad.html.

[RFC1361] Mills, D., "Simple Network Time Protocol (SNTP)", RFC 1361, August 1992.

[RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC3095] Bormann, C., Burmeister, C., Degermark, M., Fukushima, H., Hannu, H., Jonsson, L-E., Hakenberg, R., Koren, T., Le, K., Liu, Z., Martensson, A., Miyazaki, A., Svanbro, K., Wiebke, T., Yoshimura, T., and H. Zheng, "RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed", RFC 3095, July 2001.

[RFC3686] Housley, R., "Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)", RFC 3686, January 2004.

[RFC3768] Hinden, R., "Virtual Router Redundancy Protocol (VRRP)", RFC 3768, April 2004.

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", BCP 107, RFC 4107, June 2005.

[RFC4118] Yang, L., Zerfos, P., and E. Sadot, "Architecture Taxonomy for Control and Provisioning of Wireless Access Points (CAPWAP)", RFC 4118, June 2005.

[RFC4309] Housley, R., "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)", RFC 4309, December 2005.

[RFC5282] Black, D. and D. McGrew, "Using Authenticated Encryption Algorithms with the Encrypted Payload of the Internet Key Exchange version 2 (IKEv2) Protocol", RFC 5282, August 2008.

[RFC5295] Salowey, J., Dondeti, L., Narayanan, V., and M. Nakhjiri, "Specification for the Derivation of Root Keys from an Extended Master Session Key (EMSK)", RFC 5295, August 2008.

[RFC6054] McGrew, D. and B. Weis, "Using Counter Modes with Encapsulating Security Payload (ESP) and Authentication Header (AH) to Protect Group Traffic", RFC 6054, November 2010.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011.

Author's Address

David A. McGrew
Cisco Systems, Inc.
13600 Dulles Technology Drive
Herndon, VA 20171
US

Phone: (408) 525 8651
Email: mcgrew@cisco.com
URI: http://www.mindspring.com/~dmcgrew/dam.htm