Low power wireless scenarios and techniques for saving bandwidth without sacrificing security

David McGrew
mcgrew@cisco.com

Cisco Systems
April 1, 2015

Abstract. Many constrained interconnected devices rely on wireless communication and battery power, and thus need to minimize the amount of data that they send. The energy cost of transmitting and receiving data is often several times that of the cryptographic processing. Therefore, when considering cryptographic schemes to protect constrained wireless communications, it is essential to consider the power consumed to communicate the additional data required by the scheme. In this note, we review some constrained wireless scenarios and their characteristics, consider the requirements imposed by bandwidth constrains, and present a new approach to minimizing the additional data required by a cryptographic scheme. This technique can be used with different cryptographic primitives, and it has attractive security properties. It works by combining authentication and anti-replay protection and avoiding the need for a separate nonce or initialization vector.

1 Introduction

Wireless communication is ubiquitous, and will be predominant in 'Internet of Things' scenarios in which physical components containing embedded electronics can communicate with other devices or information systems, such as those of an operator or manufacturer. Many of these devices are powered by batteries, and thus need to conserve power. Several standards have emerged to address these needs, most notably the IEEE 802.15.4-2003 standard for Low Rate Wireless Personal Area Networks. The Internet protocol has been adapted to run over 802.15.4 through the IETF IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) standard [2]. Another important protocol is the LoRaWAN Low Power Wide Area Network (LPWAN), which addresses the needs for mobility, localization services, and wide ranging communication for 'smart things'.

Communication security is needed for many of these devices, especially in scenarios where communications traverses an untrusted domain such

as the Internet or a physically unprotected area. In some cases, link layer security is used, to secure all of the communications on a wireless WAN or WPAN. In other cases, transport or network level security is needed, in order to secure communications across the Internet. The former cases will require dedicated link layer security protocols integrated into the systems (like 802.11i and 802.15.4 security), and the latter need to use higher layer protocols like the conventional Internet standards (TLS, ESP, SRTP). Importantly, in some scenarios both link layer and higher layer security may be needed.

Battery power must be conserved on these devices; thus it is a major goal for their cryptographic implementations to minimize power consumption. This consideration has been paramount in the design of symmetric cryptographic primitives for constrained devices. However, there is another important cost that must be considered: the cost of transmitting and receiving, over a wireless interface, the data overhead associated with the cryptographic protocol. The *per-message overhead* of a communication security protocol is the number of bytes that must be transmitted to send a plaintext message, over and above the number of bytes in that plaintext message. The *overhead* of a protocol is the per-message overhead, averaged over many messages for typical traffic. For instance, if a protocol adds a 16-byte Initialization Vector (IV), an average of eight bytes of padding, and a 16-byte Message Authentication Code, then it has an overhead of forty bytes.

The goal of minimizing overhead is especially important for two reasons:

- the transmission of data has a very significant power cost, which typically exceeds the cost of cryptographic processing, and
- the shorter that a message is, the more likely that it is to avoid interference and thus be successfully received. Thus cryptographic schemes with higher per-message overhead are more likely to end up being retransmitted, and thus consume even more power.

Importantly, studies about the use of cryptography to protect wireless communications reveal that the power cost of transmission and reception of data typically far outweighs the power cost of the cryptographic algorithms. For instance, Seys and Preneel show that it takes 310 microjoules to encrypt a 128-bit block with the AES-128 cipher, while it takes 1408 microjoules to transmit it [5].

There are several sources of overhead in communication security protocols. Encryption algorithms typically have a nonce or Initialization Vector (IV), and block cipher padding may also cause the ciphertext to be

larger than the plaintext. Message Authentication Codes (MACs) are used to protect authenticity, and they add overhead as well. Most communication security protocols also provide an anti-replay service, usually by incorporating a sequence number in each message, and including that sequence number in the part of the message that is authenticated. We refer to this as the Authenticated Sequence Number (ASN) method; its use is widespread. Each ASN message contains a sequence number and an authentication tag generated by a message authentication code. (In some cases, authenticated encryption is used; these methods also generate authentication tags.)

In this note, we introduce an approach to communication security with less overhead - an authenticated encryption method that incorporates replay protection, and does so in a way that overloads a sequence number so that it can provide replay protection, act as a nonce, and also provide message authentication. We call this method *Authenticated Encryption with Replay prOtection*, or AERO. This new technique is well suited for secure low power wireless communication, since the computational cost of the cryptography is significantly lower than the cost of transmitting and receiving data.

2 Plaintext sizes

When the plaintext messages being used are small, the per-message overhead is important. The plaintext messages used in many low-power wireless applications are relatively small, as the designers of those systems aim to minimize the use of battery-operated radios. Most of the low-power wireless protocols have maximum packet sizes that are relatively small. In this section we review several of these protocols and applications.

Most current 802.15.4g implementations limit the 6LoWPAN payload to 800 bytes. The Distributed Network Protocol DNP3 (IEEE Std 1815-2010) has a maximum payload size of 292 bytes. The LoRaWAN protocol supports packets up to 255 bytes; however, at lower data rates the payload sizes are more limited. LoRaWAN defines several low data rate operating modes for which the maximum payload sizes are 11, 15, 129, and 242 bytes.

The American National Standard for Protocol Specification for Interfacing to Data Communication Networks, ANSI C12.22, has plaintexts with an average size ranging from 64 to 600 bytes.

3 Cryptographic background

An *arbitrary length permutation*, or ALP, is a reversible function that accepts an input with an arbitrary length. It is a permutation in the sense that, since it is reversible, it defines a permutation of all of the the elements that have a particular length. A *pseudorandom arbitrary length permutation* is an arbitrary length permutation that also accepts a secret key as an input, and has the property that, if the key is chosen uniformly at random, then it it is hard to distinguish from an arbitrary length permutation that was chosen uniformly at random from the set of all such functions. In mathematical notation, a pseudorandom ALP that has a k-bit key and accepts inputs with lengths between $n_{\min}$ and $n_{\max}$ consists of an encryption function $\mathcal{E}_{\text{ALP}} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ and $D_{\text{ALP}} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ such that $\mathcal{E}(K, \mathcal{D}(K, P)) = P$ for all $K \in \{0,1\}^k$, for all $P \in \{0,1\}^n$ and all $n \in [n_{\min}, n_{\max}]$. When an ALP is used for encryption, P is the plaintext, and $C = \mathcal{E}(K, P)$.

An *arbitrary length permutation with associated data*, or ALPA, is an arbitrary length permutation that accepts an additional input, called the associated data. The output of the ALPA is a reversible function of the plaintext (but not the associated data). For each particular value of the associated data input, the ALPA is a different function of the plaintext. The associated data can have any length. A pseudorandom ALPA is a is an ALPA that also accepts a secret key as an input, and has the property that, if the key is chosen uniformly at random, then it it is hard to distinguish from an ALPA that was chosen uniformly at random from the set of all such functions. A good way to think of a pseudorandom ALPA is that each distinct value of the associated data selects a distinct pseudorandom ALP. In mathematical notation, a pseudorandom ALPA that has a k-bit key and accepts inputs with lengths between $l_{\min}$ and $l_{\max}$ consists of an encryption function $\mathcal{E}_{\text{ALPA}} : \{0,1\}^k \times \{0,1\}^n \times \{0,1\}^a \to \{0,1\}^n$ and $\mathcal{D}_{\text{ALPA}} : \{0,1\}^k \times \{0,1\}^n \times \{0,1\}^a \to \{0,1\}^n$ such that $\mathcal{E}(K, \mathcal{D}(K, P, A), A) = P$ for all $K \in \{0,1\}^k$, for all $P \in \{0,1\}^n$ and all $n \in [n_{\min}, n_{\max}]$ and $a \in [a_{\min}, a_{\max}]$.

3.1 Authenticated encryption with associated data

An algorithm that provides authenticated encryption with associated data, or AEAD, is an encryption method that authenticates the plaintext, and also authenticates some associated data [3]. An AEAD scheme consists of an encryption function and a decryption function. The latter function either returns the plaintext, if the authentication check is passed,

or otherwise returns a symbol FAIL that indicates an authentication failure.

AERO is a stateful authenticated encryption method. There has been little formal work in this area, though there has been work on stateful message authentication, in particular as a way to improve the security of message authentication codes that would otherwise be vulnerable to repeat forgery attacks.

3.2 Anti-replay protection

In a replay attack, the attacker records one or more messages as they are transmitted, then later injects one or more of these messages into the communication channel. If there is no anti-replay protection that is provided, then the receiver(s) will accept these messages, regardless of whether or not they are encrypted and/or authenticated, since they are valid messages created with the appropriate secret key. A replay attack allows an attacker to manipulate the post-decryption plaintext. Thus, anti-replay protection is an essential part of communications security. Typical communication security protocols provide anti-replay service using the Authenticated Sequence Number (ASN) technique; ESP, AH, SRTP, 802.1AE, 802.11i, SSL, TLS, DTLS, and SSH all use a variant of this method. A sequence number is incorporated in each message that is sent, and that field is included in the part of the message that is authenticated. The sequence number fields of successive messages are set to successive numbers. After receiving and verifying the authenticity of a particular message, the receiver checks that the sequence number in the message is distinct from all previously accepted sequence numbers. If it is distinct, then the sequence number is accepted and the message is accepted as well; otherwise, both the sequence number and the message are rejected. If the messages are delivered in order, then the receiver need only store the last accepted sequence number, and compare this value to the sequence number field to perform the anti-replay check. If the messages are potentially delivered out of order, then the anti-replay checking process must take this into account. An efficient way to do this is to have the receiver store a bitmask M along with an integer $s_{\max}$ that represents the highest sequence number that has been accepted, with the convention that M_i is equal to one if the sequence number $s_{\max} - i$ has been accepted, and is equal to zero otherwise. The value of $s_{\max}$ is initially set to zero, and the receiver checks and advances the sequence number and bitmask as needed. The complete anti-replay checking process is detailed in Algorithm 3, and the initialization for the sender and receiver processes

are detailed in Algorithms 1 and 2, in which the length of the bitmask M is denoted as b. We denote the number of bits in the sequence number as t.

Algorithm 1 ASNsenderInit() ASN sender initialization)

Set $s_{\max}$ to zero.

Algorithm 2 ASNreceiverInit() (ASN receiver initialization)

Set $s_{\max}$ to w.
Set the bitmask M to the all-zero value.

3.3 Implicit sequence numbers

To reduce the overhead, some communication security protocols use the idea of an *implicit sequence number*, in which all or part of the sequence number are not transmitted. The receiver estimates all or part of the sequence number of a message, based on the ordering of the message. The implicit part of the sequence number is included in the authenticated part of the message, and if the receiver's estimate of the sequence number is correct, then the authentication verification step will pass, and otherwise it will fail. If the authentication step passes, then the anti-replay checking can proceed as usual. This process is used by SRTP and ESP.

Implicit sequence numbers work well in some scenarios, but not in others. If there are multiple receivers, then a receiver that starts receiving a stream of messages after the transmission has started will not know the current value of the implicit sequence number. That value will need to be communicated to the receiver, or the receiver will need to determine that value through trial and error. Both methods add undesirable complexity. In addition, implementation complexity of implicit sequence numbers makes it undesirable for implementation in hardware; both 802.11 and 802.1AE avoid this technique.

4 Authenticated encryption with replay protection

In this section we describe AERO, a new technique for providing encryption, message authentication and anti-replay protection, which has

Algorithm 3 ASNcheck(s) (ASN checking of the sequence number s)

Input: Sequence number $s \in [0, 2^t - 1]$
Output: ACCEPT or REJECT

```
if s = 0 then
    return REJECT
end if
if s > s_max then
    d ← s − s_max
    if d < b then
        M ≪ d
        M[1] ← 1
    else
        M ← all zeros
        M[1] ← 1
    end if
    s_max ← s
    return ACCEPT
end if
d ← s_max − s
if d ≥ b then
    return REJECT
end if
if M[d] = 1 then
    return REJECT
end if
M[d] ← 1
return ACCEPT
```

less overhead than the conventional ASN with Authenticated Encryption method. This technique makes use of an encryption algorithm that is an pseudorandom ALPA.

AERO works as follows. The sender maintains an t-bit sequence number s, which is initially equal to zero. The receiver maintains the bitmask M and highest accepted sequence number $s_{\max}$ as used in the ASN method; the AERO algorithms share the value $s_{\max}$ with Algorithms 1, 2 and 3, and those algorithms are used as subroutines by the AERO algorithms. The receiver also maintains a t-bit value r that holds the last rejected sequence number.

To encrypt a plaintext P, the sender concatenates s with P, then encrypts the result with an ALPA cipher using a secret key K, using the associated data A provided with the plaintext. The resulting ciphertext C is the entire message used in the security protocol; it incorporates encryption, message authentication, and anti-replay sequencing data. The AERO encryption and decryption operations are defined in Algorithms 6 and 7. These algorithms use parameters v and w, which define the length of the 'windows' of sequence numbers that will be accepted, even though they are slightly larger than expected.

Algorithm 4 AERO encryption initialization.

Input: Key $K' \in \{0,1\}^k$
$K \leftarrow K'$
ASNsenderInit()

Algorithm 5 AERO decryption initialization.

Input: Key $K' \in \{0,1\}^k$
$K \leftarrow K'$
ASNreceiverInit()
$r \leftarrow 0$

We call the value s in Algorithm 7 the *candidate sequence number*. AERO decryption incorporates a way to re-synchronize a sender and a receiver in case w or more messages are lost in transit. The logic that compares the current candidate sequence number s to the last rejected candidate sequence number r enables re-synchronization. If this re-synchronization process occurs, then at least one valid message will be

Algorithm 6 AERO encryption.

Input: Plaintext $P \in \{0,1\}^n$, associated data $A \in \{0,1\}^a$
Output: Ciphertext $C \in \{0,1\}^{n+t}$

$C \leftarrow \mathcal{E}_{\mathrm{ALPA}}(K, P\|s, A)$
$s \leftarrow s + 1$
return C

Algorithm 7 AERO decryption.

Input: Ciphertext $C \in \{0,1\}^{n+t}$, associated data $A \in \{0,1\}^a$
Output: Plaintext $P \in \{0,1\}^n$ or the symbol FAIL

$P\|s \leftarrow \mathcal{D}_{\mathrm{ALPA}}(K, C, A)$
if $s > s_{\max} + w$ **then**
  if $s - r > v$ or $s \leq r$ **then**
    $r \leftarrow s$
    return FAIL
  else
    $s_{\max} \leftarrow s$
    return P
  end if
else
  $z \leftarrow \mathrm{ASNcheck}(s)$
  if $z = \mathrm{REJECT}$ **then**
    $r \leftarrow s$
    return FAIL
  else
    return P
  end if
end if

rejected by this algorithm. This is undesirable; however, it is acceptable, considering that a loss of at least w messages has occurred.

5 Security

In this section, we assume that the ALPA scheme used in AERO is sound, i.e. for any choice of secret key, an attacker cannot distinguish it from a ALPA chosen uniformly at random. (We do not yet provide a proof of security using a reduction argument that shows that if AERO can be broken, then the ALPA scheme can be broken, though this is clearly possible to do.) For both AERO and ALPA, we consider the attack model in which the adversary is able to submit chosen plaintexts and chosen ciphertexts, and do so adaptively.

The soundness of AERO as an encryption method is easy to see; it follows directly from the soundness of the ALPA encryption. Because each invocation of $\mathcal{E}_{\mathrm{ALPA}}$ includes a distinct value of the sequence number, each AERO ciphertext is distinct. Thus, AERO encryption meets the strongest definition of confidentiality; in particular, if the same plaintext value is used in multiple invocations of $\mathcal{E}_{\mathrm{ALPA}}$, the result will be distinct ciphertext values.

The soundness of AERO authentication follows from the fact that an attacker is highly unlikely to be able to find a ciphertext that will be accepted by the AERO decryption algorithm. In a forgery attempt, an attacker submits a ciphertext C and an associated data A to $\mathcal{D}_{\mathrm{AERO}}$ that does not match any ciphertext C' and associated data A' that have been output or input (respectively) to $\mathcal{E}_{\mathrm{AERO}}$. (That is, we make the usual assumption that when trying to forge messages, the attacker does not replay valid messages.) Because of the soundness of $\mathcal{D}_{\mathrm{ALPA}}$, the post-decryption plaintexts are indistinguishable from random values, to an attacker who does not know the secret key. Thus, the attacker will be unable to distinguish s from random, much less control that value. Hence it is suitable to assume that, during a forgery attempt, the value of s in Algorithm 7 is distributed uniformly at random between 0 and $2^t - 1$ at the outset of the decryption process. The number of values of s that cause that algorithm to return P is $v + w + b$. Thus the probability that an individual forgery attempt will succeed is at most $(v + w + b)2^{-t}$.

The soundness of AERO anti-replay protection is easy to see: a replayed message will undergo the same anti-replay checking as the ASN method.

A denial of service attack against AERO is no more effective than a denial of service attack against ASN, assuming that the two methods have similar probability of an individual forgery attempt being successful.

5.1 Misuse resistance

A desirable property for an encryption method is that of *misuse resistance*: security should degrade gracefully if the method is used improperly. AERO is an authenticated encryption method that strongly resists misuse. It does not require a nonce input, which makes it simpler to use. In these respects, it addresses the security concerns that have been expressed by the academic community about current authenticated encryption techniques such as Galois/Counter Mode (GCM) [4, 1].

In some scenarios, such as virtual machine environments, it may be impossible to guarantee that the sequence number used in the encryption processes is distinct. In particular, a virtual machine containing a secret key and sequence number may be cloned or duplicated, so that multiple virtual machines are active with the same secret key and sequence number. To protect against this situation, the encryption process can include a random value that is discarded by the receiver. As long as the encryption process uses a good random source for this value, strong confidentiality will ensue. Legitimate messages output by the encryption process may be rejected by the decryption process; note however that this is an communication/system/interoperability problem and not a security problem.

6 Overhead comparison

The overhead of AERO is t bits, and a single key can be used to send 2^t messages. The ASN method is characterized by the number of bits x in its authentication tag, and the number of bits c in its sequence number. The overhead of this technique is $x + c$, the forgery probability is at most 2^{-x}, and a single key can be used to send 2^c messages.

To show the benefits of the new technique, we consider the situation in which t, x, and c are chosen so that the two methods have identical forgery probabilities, in which case $t = x + \lg(v + w + b)$. In practice b is usually set to 32 (for most protocols) or 128 (for SRTP), and it is reasonable to set $w = b$ and $v = 1$. In these cases, t is larger than x by no more than eight bits. The new method has less overhead by a factor of $\alpha = c - \lg(v + w + b)$.

For low-power wireless, AERO can save several bytes, which is quite significant for the small frame sizes typical of that domain. For example, the 802.15.4 use of AES-CCM for link-layer wireless encryption uses a $c =$ 32-bit sequence number (the Frame Counter field) and an authentication tag of $t \in \{32, 64, 128\}$ bits. Using AERO instead would save 24 bits (three bytes), where frame sizes are between 10 and 108 bytes, for a bandwidth saving between 3% and 30%.

There are more modest but still worthwhile savings even for conventional WiFi. The bandwidth savings of AERO on Internet traffic over 802.11i can be estimated by using the IMIX estimate: 58% of packets have 40 bytes, 33% of packets have 576 bytes, and 8% of packets have 1500 bytes. Assuming that the plaintext packets have the IMIX distribution, the distribution of the lengths of 802.11i AES-CCMP protected frames that are transmitted over the wireless link layer is as follows: 58% of packets have 60 bytes, 33% of packets have 596 bytes, and 8% of packets have 1520 bytes. This gives an average frame size of 353 bytes. In contrast, AERO would have overhead of 14 bytes instead of 20 bytes, giving a frame-length distribution of: 58% of packets have 54 bytes, 33% of packets have 590 bytes, and 8% of packets have 1514 bytes. This gives an average frame size of 347 bytes. Thus, AES-CCMP has nearly 2% more bandwidth overhead than AERO would have. Thus AERO has a modest but worthwhile savings even when applied to Internet traffic over WiFi.

The ESP protocol, which protects data within the IPsec framework, has $\alpha = 24$ bits (three bytes). However, ESP also includes an Initialization Vector in each packet, for another 64 bits (for the CTR, GCM, CCM, and GMAC modes of operation) or 128 bits (for the CBC mode of operation). Thus, AERO can improve on ESP by at least 88 bits (11 bytes).

It is less easy to quantify the reduction in overhead for Secure RTP, because that protocol uses implicit sequence numbers and sometimes communicates the entire sequence number. However, there is a significant benefit for using AERO in SRTP: it makes it unnecessary to maintain the implicit sequence number between the sender and receiver. AERO significantly simplifies SRTP for multi-party communication such as media bridges, while having very little additional data expansion. In SRTP, authentication tags are sometimes as short as four bytes. The choice of $t = 32$ for AERO may be undesirably small, because it would result in a forgery probability of about 2^{-24} and would restrict the number of packets that could be encrypted with a single key to $2^{32} - 1$. It seems safer to use $t = 40, 48, 64$ if bandwidth is highly constrained, and a larger value otherwise.

APRP ciphers are typically slightly more expensive than conventional encryption techniques, and thus the advantage of reduced communication cost must be weighed against the potential disadvantage of increased power cost.

The detailed systems advantage of this bandwidth savings can be found as follows. We let x_{CCMP} denote the amount of power used to transmit a byte of data, and c_{CCMP} denote the amount of power used to cryptographically protect a byte of data, with the CCMP algorithm. We also let x_{AERO} and c_{AERO} denote the comparable values for the AERO algorithm in 802.11i. Denoting the fractional bandwidth savings of AERO over CCMP as γ, we have $x_{\mathrm{CCMP}} = (1+\gamma)x_{\mathrm{AERO}}$ and AERO uses less overall power than CCMP whenever

$$\gamma x_{\mathrm{CCMP}} > c_{\mathrm{AERO}} - c_{\mathrm{CCMP}}$$

or equivalently

$$\frac{c_{\mathrm{AERO}}}{c_{\mathrm{CCMP}}} < \gamma \frac{x_{\mathrm{CCMP}}}{c_{\mathrm{CCMP}}} + 1. \tag{1}$$

For example, if the power needed to transmit a byte is one hundred times the power needed to CCMP-protect a byte, then Equation 1 becomes $c_{\mathrm{AERO}}/c_{\mathrm{CCMP}} < 3$.

If AERO is used in a higher-layer protocol such as IPsec, TLS, DTLS, or SSH, it reduces the overhead of those protocols, and thus makes better use of whatever wireless links it may be traversing. In this sense, AERO can be said to be wireless-friendly even when it is used in protocols that are above the link layer.

7 Performance

ALPA schemes have worse performance than other encryption schemes, because they require at least two serialized passes over the plaintext input (during encryption) and the ciphertext input (during decryption). This means that plaintext and ciphertext must be stored in a temporary buffer during the encryption and decryption operations. In a very high throughput system, this is only possible if the size of the messages is limited. Thus AERO is probably unsuitable for high speed applications such as 802.1AE ethernet encryption, but it is perfectly suitable for many other applications.

For wireless systems, the increased cost of the AERO computation is typically worth the decreased costs in transmission and reception that follow from the lower data overhead.

We anticipate that AERO will be advantageous for wireless protocols whenever it is possible to use a dedicated encryption circuit, since such circuits minimize power consumption. We also expect AERO to excel for long-range wireless, where transmission costs are high. However, it may not be advantageous for short-range transmissions where no dedicated circuit is available. More data is needed before a detailed comparison and analysis can be done.

8 Conclusions

Communication security for low power wireless networks should minimize the data overhead. This imperative is especially true when plaintext sizes are small, and when security is needed at multiple layers in the network stack. It is possible to reduce the bandwidth overhead required for communication security by combining anti-replay protection into authentication encryption. Low-power wireless devices (e.g. 802.15.4) could save significant power by reducing their transmission and reception costs up to 30% or more. The scheme that we presented also has the benefit of AERO that it provides misuse resistance, i.e. it is robust in the face of applications that cannot properly manage nonces. This fact offers significant simplifications to protocols like SRTP, by relieving them of the need to manage and coordinate an implicit sequence number. This is especially appealing for multiparty applications such as conference bridging.

References

1. Morris Dworkin. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. National Institute of Standards and Technology (NIST) Special Publication SP 800-38C, 2004.
2. J. Hui and P. Thubert. Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. RFC 6282 (Proposed Standard), September 2011.
3. D. McGrew. An Interface and Algorithms for Authenticated Encryption. RFC 5116 (Proposed Standard), January 2008.
4. David A. McGrew and John Viega. The security and performance of the Galois/counter mode (gcm) of operation. In Anne Canteaut and Kapalee Viswanathan, editors, *Progress in Cryptology - INDOCRYPT 2004: 5th International Conference in Cryptology in India*, volume 3348 of *Lecture Notes in Computer Science*, pages 343–355, Chennai, India, December 20–22, 2004. Springer, Berlin, Germany.

5. Stefaan Seys and Bart Preneel. Power consumption evaluation of efficient digital signature schemes for low power devices. In *2005 IEEE International Conference on Wireless And Mobile Computing, Networking And Communications, WiMob 2005, Montreal, Canada, August 22-14, 2005, Volume 1*, pages 79–86, 2005.