# Recommendations for IPsec configuration on homenet and M2M devices

Daniel Migault, Tobias Guggemos, Daniel Palomares, Aurelien Wailly, Maryline Laurent, Jean-Philippe Wary

# Recommendations for IPsec Configuration on Homenet and M2M devices

Daniel Migault
Ericsson

Tobias Guggemos
LMU Munich

Daniel Palomares
Orange

Aurelien Wailly
Orange

Maryline Laurent
Institut Mines-TELECOM,
UMR CNRS 5157 SAMOVAR

Jean Philippe Wary
Orange

## ABSTRACT

Although there is a strong need to deploy secure communications in home networks and for Machine-to-Machine (M2M) environment, to our knowledge the impact of authenticated encryption migration has not been evaluated yet. As the security performance issue is especially critical for wireless environment, this paper measures the effect of the security settings on the Quality of Service (QoS) for encrypted communications in a home network environment. Security settings include different configurations of IPsec tested over several hardware platforms. The QoS is evaluated based on CPU time and elapsed time for downloading different sized files.

## 1. INTRODUCTION

The Internet community has recently realised the effects of intrusive Internet surveillance on the end user's privacy. This phenomenon has been designated by Stephen Farrell et al. [6] and the whole Internet community [7] as pervasive monitoring. This resulted in a number of consequences [24], and one of them is a clear shift for strong encryption.

In the area of encryption, Authenticated Encryption like AES-GCM [16] is highly encouraged as it is considered as the most efficient NIST standard scheme for Authenticated Encryption. In addition, several efforts have been provided by vendors to enhance the AES-GCM performances [8, 10, 11]. For example, Intel provides specific instructions that optimize AES-GCM like AES-NI [10] and PCLMULQDQ [11], and publish patches for opensource software like OpenSSL [12].

Although there is a strong need to deploy secure communications in home networks and for Machine-to-Machine (M2M) environment, to our knowledge the effects of authenticated encryption migration has not been evaluated yet.
In fact, appliance communications in a home environment are likely to leak information with privacy concerns. This information range from your personal photos between your Network Attached Storage (NAS) to your phone, to the power consumption provided by your sensor that may reveal your activity, your presence, the type of appliance used among others.
On the other hand, not all devices can provide AES-GCM secure communications, as mostly recent equipments has been designed for AES-GCM. This is especially true in home network environment or in machine-to-machine communications where probes have been deployed out-of the box with little attention provided for evolutions and possible upgrades. In addition, AES-GCM optimization has mostly been optimized over heavy load, which does not match probes utilization.

This paper measures how the effect of the security settings on the Quality of Service (QoS) for encrypted communications in a home network environment. Security settings include different configurations of IPsec tested over several hardware platforms. The QoS is evaluated based on CPU time and elapsed time for downloading different sized files. This paper is addressing the following questions:

- For a given hardware how can I chose the appropriate security settings that have the lowest impact on QoS?
- As a Customer Premises Equipments (CPE) or Set-Top Boxes (STB) designer how should I appropriately design the hardware to support authenticated encryption?

The paper is organized as follows. Section 2 gives an overview of some related works. Section 3 describes the devices as well as the performance measurements methodology. Section 4 compares the effect of the AES modes on the QoS. Section 5 evaluates the effect of various IPsec configurations on the QoS. Section 6 measures the effect of power saving modes. Finally, section 7 compares IPsec vs TLS and section 8 concludes this work.

## 2. RELATED WORK

A. Hoban [13] introduces some tests done by Intel using IPsec in tunnel mode under loaded conditions. Although our paper does not concern load tests, it focuses on singular

tests addressing homenet, IoT and M2M perspectives. A. Hoban performed measurements among several AES implementations to evaluate AES-NI and AES-GCM crypto plugin. The article analyzes various throughputs in Mbps and emphasizes a benefit of 400% when the AES-GCM crypto plugin is enabled. In addition, the results obtained in [13] concerning original AES-NI vs AES-NI disabled, are similar to ours in section 4.3.

In [16], McGrew & Viega demonstrated that modes of authenticated encryption GCM and CCM are more efficient than combining an encryption function (AES-CTR, AES-CBC) and a MAC algorithm (e.g. HMAC-SHA-1). This is indeed the trend today for getting higher performance and security.

Granjal et al. [9] explain why IPsec is an interesting alternative to secure IoT communications. The paper focuses on the evaluation of different encryption algorithms, and also explains the need for IPsec as 6LoWPAN [15] requires IP layer security. Other materials like 6LoWPAN-IPsec [21] and RoHC [23] define compression to reduce IPsec overhead to fit limited MAC-frames as in IEEE 802.15.4. Such compression reduces communication costs for IoT devices, where data transmission is usually much more resource consuming than calculation.

Current investigations address application layer security for IoT devices. The IETF DTLS In Constrained Environments (DICE) working group focuses on DTLS for UDP-layer security on sensors. In addition, Raza et. al. [22] describe how the packet overhead of the DTLS protocol can be reduced significantly.

Our measurements confirm Granjal et.al's works [9], as the performances of IPsec are better or similar to TLS as long as the implementation does not refer to highly optimized application cryptographic tool-kit for some specific processor.

# 3. PLATFORM DESCRIPTION

Throughout this paper, different processors are used in several performance tests. These processors are listed below:

- *E6410* [5] (Intel(R) i5 CPU M540 @ 2.53 GHz) a mobile Westmere based architecture also known as Arrandale. This processor has been tested on Mobile Desktop and delivers the AES Instruction set (AES-NI for Advanced Encryption Standard - New Instructions) [10] and Carry Less MULtiplication (PCLMULQDQ) [11], as well as stitching mechanisms [8] used to speed up cryptographic operations. This type of hardware represents the upper bound of low powered devices.

- *NC10* [18] (Intel(R) Atom CPU N270 @ 1.6GHz [17]) a mobile Bonnell architecture designed for ultra low voltage. This device represents either evolved M2M homenet devices or homenet devices the end user interacts with.

- *RPi* [20] (Rasberry Pi ARM 1176JZF-S (ARMv6k) 700 MHz [2]) a one core processor. This device typically represents low powered homenet devices used for M2M, as Customer Premises Equipment (CPE) or Set-Top-Unit (STU) provided by ISPs or vendors are likely to embed this kind of technology.

For measuring performances related to security functions only, the testing platform is limited to Ethernet connectivity between devices. In addition, considering end-user requirements, performances are evaluated considering the following parameters: 1) the download elapsed time, which is the time between invocation and termination of the command (i.e. to download the entire file), and 2) the resources involved for the downloading, which are evaluated by considering the time spent within the CPU during the linux command `time`. Note that the CPU time considers both the *user CPU time* spent executing instructions of the calling process and the *system CPU* time spent in the system while executing tasks on behalf of the calling process.

# 4. AES MODES WITH IPSEC

This section provides inputs to adequately select a cryptographic mode of AES according to 1) the packet size of the IP payload, 2) the number of cores available on the hardware, 3) the hardware —*E6410* , *NC10* , *RPi* —and 4) AES implementations: software AES implementation (sAES) vs. hardware AES implementation (AES-NI).

This paper only considers AES [1] for encryption support as recommended by the NIST (National Institute of Standards and Technology). AES is provided by multiple vendors as an AES hardware embedded within the CPU, such as AES-NI [10]. In this paper, AES is tested in the following modes: 1) CBC, 2) CTR, 3) CCM [3] and 4) GCM [4, 16]. CCM and GCM use AES-CTR for encryption, while CCM (resp. GCM) uses CBC-MAC (resp. GMAC) for authentication. AES-CBC and AES-CTR only supports encryption function and refer to the external HMAC-SHA1-96 mechanism for authentication. Please refer to table 1 to understand the AES modes considered throughout this paper.

| Notation | CTR | CBC | CCM | GCM |
|---|---|---|---|---|
| Encryp. | AES-CTR | AES-CBC | AES-CTR | |
| Auth. | HMAC-SHA1-96 | | CBC-MAC | GMAC |

**Table 1: Encryption and Authentication for each AES mode**

## 4.1 Mode vs. Packet Size

This section measures how significant is the packet size on the choice of the AES mode. IPsec is used with transport mode. Note that with IPsec the encrypted data are independent of the MTU, as IPsec fragmentation occurs over the encrypted payload. Thus, reducing the MTU size increases the number of IP headers transmitted as well as the number of networking packet processing, but encrypted data remain the same.

Figures 1 and 2 respectively depict the elapsed time and the CPU time for *E6410* , *NC10* and *E6410* with AES-NI and sAES. The ratio over CBC is represented in order to easily compare the different AES modes where the horizontal line with elapse time equal to 1 means symbolizes the CBC mode. Note that CPU time represents both the kernel-land and user-land CPU time. Although it is not represented in

| MTU | 180 | 200 - 1000 | 1000 - 1500 |
|---|---|---|---|
| **Soft AES** | CBC | CTR | CBC |
| **AES-NI** | CBC | CBC (CPU) / CTR (elapsed) | |

**Table 2: Encryption recommendations for different MTUs**

the figure, the user-land CPU time is marginal and is always less than 10%.

With sAES, the elapsed time in figure 1 and CPU time in figure 2 clearly show two groups of modes: 'CCM, GCM' and 'CTR, CBC'. The first group requires from 20% up to 40% more time for both elapsed and CPU time. The difference between both groups is that 'CTR, CBC' authenticates with HMAC-SHA1-96 over the whole payload, while 'CCM, GCM' authenticate on a per block basis. This increases the number of operations. For 180 byte MTU, there is no clear advantage between CBC and CTR and performances depend on the CPU. However, for small MTUs, CTR seems to provide better performances, probably because CTR does not have blocks whereas CBC has 128 bits blocks. As a result, CBC has to wait to fill in the blocks with incoming packets before proceeding to AES decryption. 'CCM, GCM' are based on CTR and are more sensitive to MTU than CTR for large MTU. When cryptographic operation costs are reduced with AES-NI —see figures 1b and 2b —all CTR-based modes 'CTR, CCM, GCM' have similar behaviors to CBC. This shows that the MTU variations have an effect on cryptographic operations. In fact, 'CCM, GCM' have additional cryptographic operations than CTR, making the operation susceptible to be interrupted. In addition, with larger packets, context switching takes more time, which makes 'CCM, GCM' with software implementation more sensitive to MTU variations.

As summed up in table 2, although CBC and CTR have similar performances, CBC performs better for 180 bytes packets, CTR performs better for packets smaller than 600 bytes, and CBC performs better for larger packets.

With AES-NI, CBC overcomes the other modes by a factor of 2 to 3 for $180-200$ bytes packets. For larger packets, CTR provides the fastest downloading, CBC and CCM are similar but slower than CTR, and GCM has the worst performances (see figure 1b). For CPU in figure 2b, CBC is more efficient, but for packets greater than 800 bytes, CTR has similar performances. With AES-NI, CCM clearly performs better than GCM, at least in Linux kernel. The reason why CCM outperforms GCM is most probably because the authentication is optimized for CCM using a CBC-MAC, whereas GMAC needs a memory extension to optimize GCM (especially to support the polynomial complexity of GCM-MAC).

*We recommend using AES-CBC and AES-CTR combined with HMAC-SHA1-96 over CCM and GCM, and this for devices equipped with hardware acceleration or not. Note that CBC has* 128 *bit block size which adds overhead over CTR. For this reason, we recommend CTR for small size data.*

## 4.2 Mode vs. Multi Cores

This section measures how cryptographic modes take advantage of multiple cores. Elapsed and CPU time are measured with 4 cores on *E6410* and 2 cores on*NC10* , AES and AES-NI. Figures 3 and 4 depict the ratio over a single core. Multi cores and parallelism can be used at two different levels. First, different IPsec packets can be treated independently by different cores. In fact, the encryption can be parallelized for AES-CTR-based modes (CTR, GCM and CCM), but not AES-CBC-based modes. Authentication can also be parallelized according to its authentication algorithm (see Table 1 and Table 2 in [19]). For combined modes (i.e. GCM and CCM) the parallelization can be performed only if both encryption and authentication are parallelizable. As CCM uses CBC-MAC for authentication, it cannot be parallelized. Parallel processing induces synchronization, so the usage of multiple cores provides an advantage when processing time overcomes synchronization time. This suggests that AES decryption is a very small operation that cannot really take advantage of multiple cores.

The effect of the MTU is well illustrated in figures 3 and 4, where small packets processed by multiple cores increase by $1.35 - 1.7$ the elapsed and CPU times. When AES-NI is involved, processing is even faster, increasing the synchronization overhead over the computation as represented by CBC in figure 3b. Figure 4 depicts the CPU time and shows that multiple core provides a slight advantage when decryption is heavy (GCM, CCM) and with a high number of cores.

Overall, multiple cores adds a huge $(1.3 - 1.7)$ overhead for small packets in both elapsed and CPU time. For larger packets, elapsed time is at least 1.1 times larger. Multiple cores may decrease by 0.85 the CPU time with heavy operations (GCM, CCM). As a result, for homenet devices, we recommend using a single core for very small packet communications.

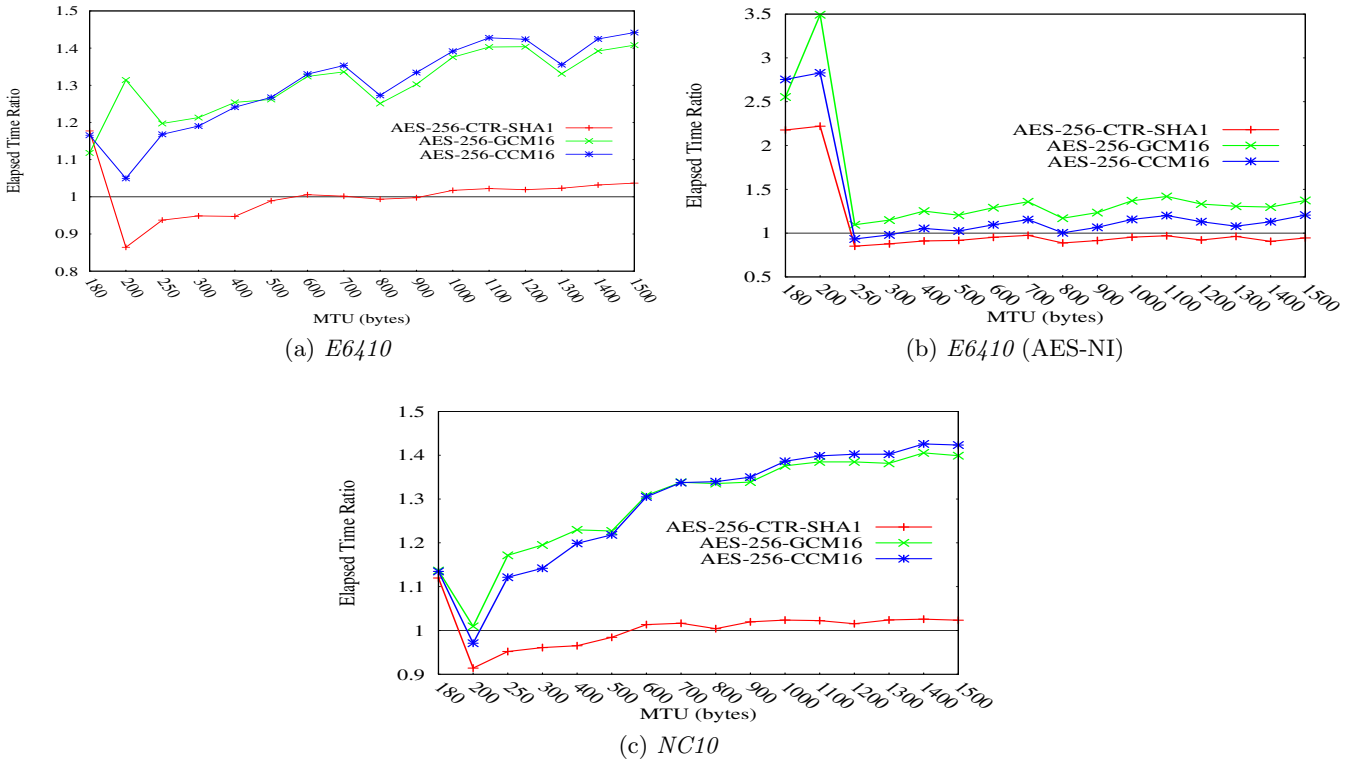
*We recommend using a single core for a singular IPsec protected communication.*

## 4.3 Mode vs. AES Implementation

This section evaluates the effect of the AES-NI when using sAES over the mode performances on an IPsec communication. IPsec is used in transport mode with one core.

For packets greater than 250 bytes, CCM takes the maximum advantage of AES-NI for the elapsed time with a 1.3 times faster download. CBC minimizes its advantage with a 1.1 times faster download. On the other hand, for packet size lower than 250 bytes, CBC maximizes the advantage of AES-NI with a 3.3 times faster download.

CPU time depicts similar results for any mode except for AES-NI, with higher impact on CBC, much more than the other modes. The main reason is that CBC does not enable parallel decryption, which means that for a given payload, CBC reflects the performance of $\frac{AES-NI}{AES}$. In other AES-CTR-based mode (CTR, CCM, GCM), each block may be decrypted separately, for example, in different threads. Parallelism induces a thread synchronization that counter balances the advantage of hardware acceleration. The reason why CPU time and elapsed time are not correlated with CBC, is that for a single packet, CBC still requires more




(a) *E6410*

(b) *E6410* (AES-NI)

(c) *NC10*

**Figure 1: AES modes (CTR, CBC, CCM, GCM) vs. MTU: Elapsed time with considered Ratio:** $\frac{EncryptionMode}{CBC-SHA-1}$

time than CTR-based modes, and elapsed time is limited by networking capability of the NIC.

The maximum benefit of AES-NI is obtained for the CBC mode in terms of resource consumption. $180 - 200$ bytes packets optimizes the download speed for CBC, whereas in the case of larger packets, it is optimized for CCM. As a result, to optimize the use of AES-NI over a IPsec communication we recommend using of CBC mode.

*We recommend using of hardware acceleration to reduce the CPU time. CCM and CBC are the modes maximizing this advantage.*

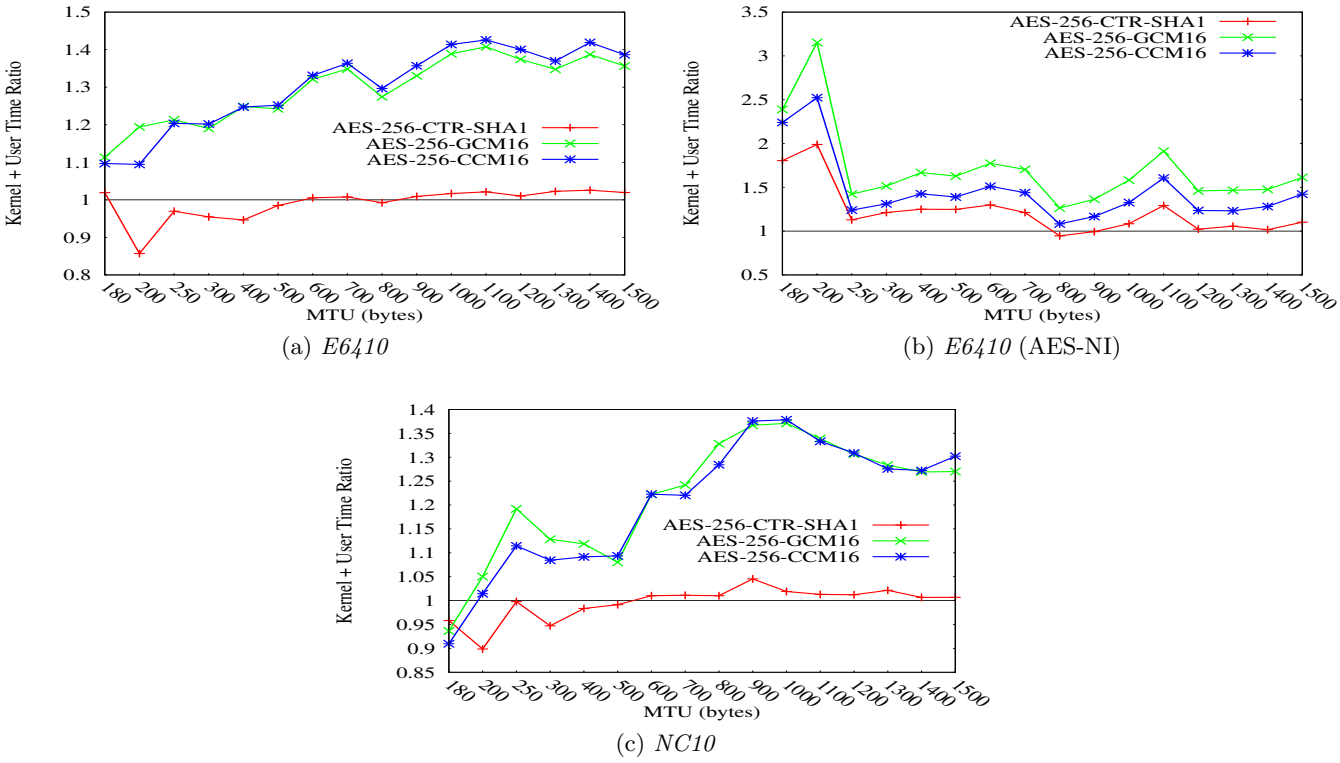
# 5. NETWORKING OVERHEAD

In this section and upcoming sections, we consider the single cryptographic mode – AES-CBC with HMAC-SHA1-96 – which characteristics have been previously given in section 4. The objective of this section is to measure the effect of different IPsec parameters over the communication. Network parameters are usually addressing different architectures and thus may not be optional. The goal of this section is to measure the effect of the architecture design on secure communications within a home network environment.

## 5.1 Network and Encryption Overhead

IPsec involves both cryptographic and networking computation. Figures 6 and 7 respectively show the IPsec encryption overhead —AES-CBC-HMAC-SHA1-96 encryption over the NULL encryption —and the IPsec networking overhead —NULL encryption over a HTTP with no IPsec —. Note that IPsec does not allow NULL encryption simultaneously with NULL authentication. As a result, the integrity protection is included in the evaluation of the IPsec network overhead itself.

With sAES, the use of multiple cores reduces the CPU time, but makes cryptographic operations longer, especially with small packets. Overall, for a single communication, multiple cores do not provide significant advantages as analyzed in section 4.2. Regarding cryptographic overhead in figure 6, *E6410* needs more CPU time to perform the cryptographic operation than the *RPi* . One possible explanation is that *E6410* makes more interruptions than the *RPi* , especially to handle incoming packets in the NIC. Note that these interruptions are not necessary to complete the crypto part, even though they are designed to handle more packets per seconds. For large packets, *E6410* is roughly 1.3 times faster than *RPi* which may results from the 3.6 times faster CPU frequency. Regarding the network overhead in figure 7, elapsed time and CPU time are highly correlated and elapsed time is affected by the CPU frequency and its interactions with the NIC. Hence, the limiting factor is the communication with the NIC (Network Interface Card), which may involve additional interruptions. Then, the higher the frequency is, the faster the packets are processed. Consequently, building ESP packets costs 1.5 time more over a non IPsec protected packet and encrypting ESP packets costs 3.75 more CPU time.




(a) *E6410*

(b) *E6410* (AES-NI)

(c) *NC10*

**Figure 2: AES modes (CTR, CBC, CCM, GCM) vs. MTU: CPU time with considered Ratio:** $\frac{Encryption Mode}{AES-256-CBC-SHA-1}$

*For homenet, IoT and M2M devices, considering CPU and elapsed time for a singular IPsec protected communication, we recommend using NULL encryption only if confidentiality is not necessary. However, we recommend using IPsec, because the network overhead is not the significant factor where time and resources are spent.*

## 5.2 IPsec Tunnel mode

The IPsec tunnel mode may be used if a device is connected to a security gateway acting as the entry point towards a trusted network. Using tunnel mode adds an IP header to the encrypted payload, resulting in a larger payload to encrypt.

Figure 8 compares elapsed and CPU time overhead of the IPsec tunnel mode over transport mode. For packet larger than 500 bytes and one CPU, the tunnel mode does not increase the elapsed time for *E6410* , *NC10* nor *RPi* . Of course the limiting factor in our test is the hardware, so this may not be true anymore in bandwidth limited networks. Using 2 or 4 cores makes the tunnel mode faster. This means that tunnel encapsulations are heavy enough operations so they can benefit from multiple cores, overcoming the synchronization overhead. Tunnel overhead is independent of the packet size as suggested by sAES implementations. With AES-NI processor, the tunnel overhead actually depends on the packet size, probably because IPsec header decryption requires an additional involuntary context switch. Note also that with AES-NI, synchronization does not overcome the additional computation.

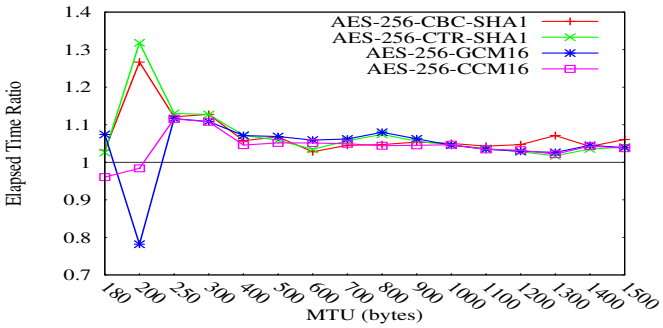
Similarly, CPU time presents an overhead over the transport mode when a single CPU is used or when using AES-NI with packets smaller than 700 bytes. For larger packets, using tunnel mode does not add additional CPU time. With sAES, CPU time takes advantage of multiple cores, with up to 0.2 times CPU time. In contrast to transport mode, implementation optimizations have been focused on tunnel mode, as it is widely used for VPNs. With AES-NI, the usage of one single core is preferred if energy consumption has a higher priority than download speed over high bandwidth environments.

Thus, using multiple cores reduces tunnel overhead with sAES. With AES-NI, multiple cores boost downloadings whereas a single core is more energy efficient. Tunnel mode may be used instead of the transport mode only under high bandwidth networks.

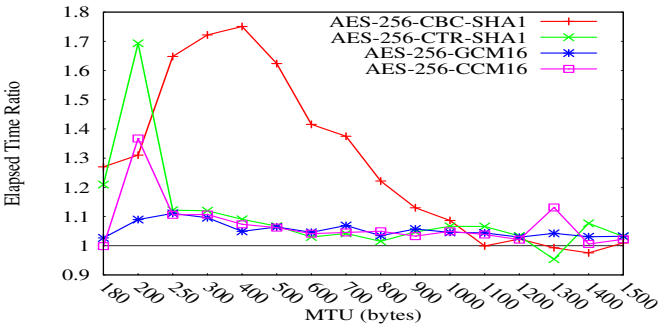
*We recommend the transport mode of IPsec when possible. Tunnel overhead is negligible for large packets, but we recommend to use one single core with AES-NI and multiple cores with sAES*

## 5.3 UDP encapsulation

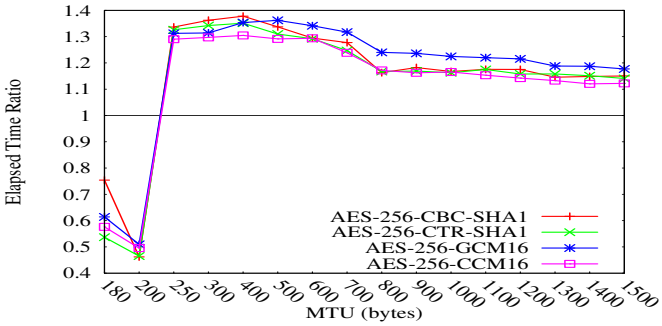
UDP encapsulation is used for NAT/middle box traversals which are not aware of IPsec traffic, and drop any non UDP or non TCP traffic. UDP encapsulation is defined for IKEv2




(a) *E6410* – Ratio: $\frac{4CPUs}{1CPU}$




(b) *E6410* (AES-NI) – Ratio: $\frac{4CPUs}{1CPU}$




(c) *NC10* – Ratio: $\frac{2CPUs}{1CPU}$

**Figure 3: AES mode vs. multiple cores: Elapsed time**

and ESP [14], and results in all IPsec and IKEv2 packets sent and received on port 4500. Typically UDP encapsulation inserts an UDP header between the IP header and the ESP payload, but leave unchanged the IP header and the ESP payload. Incoming packets are then decapsulated and processed as in the standard IPsec. The main difference between the UDP encapsulation and the IPsec tunnel mode is that the UDP encapsulation does not introduce any additional cryptographic operation whereas in the tunnel mode the inner IP header is encrypted.

Figure 9 depicts a high similarity between UDP encapsulation and IPsec tunnel mode. This shows that the limiting factor is not the encryption, but the networking operations. It also shows that the encapsulation layer has not significant impact on performance. In fact UDP encapsulation or IP encapsulation results both in pointer manipulation over the kernel structure (sk_buff).

*We recommend not to perform UDP encapsulation by default, but only when required. However, UDP encapsulation has similar effect on the performances than the tunnel mode, and thus is negligible for large packets.*

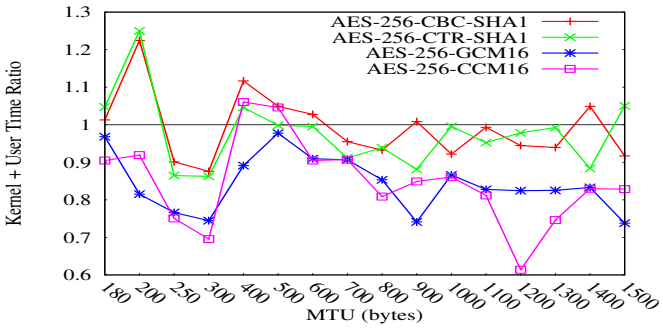
## 6. POWER USAGE OVERHEAD

This section measures the effect of different power saving policies on an IPsec communication (see Figure 10a and 10b). Power saving policies, in our case are based on frequency scaling [25]. All information are located under the `/sys/devices/system/cpu/cpu0/cpufreq` directory. *NC10* CPU is configured with a minimum frequency of 800 MHz and a maximum frequency of 1.6 GHz, and *E6410* from 1.2 GHz to 2.6 GHz. The current frequency is determined by an element called the **governor**. The governor automatically applies one of the following frequency policies: by default, the *ondemand* policy increases the frequency from minimum to maximum, the *conservative* policy increments the CPU frequency by steps, the *performance* policy keeps the CPU at its highest frequency, and finally, the *powersave* policy maintains the frequency at its minimum level.

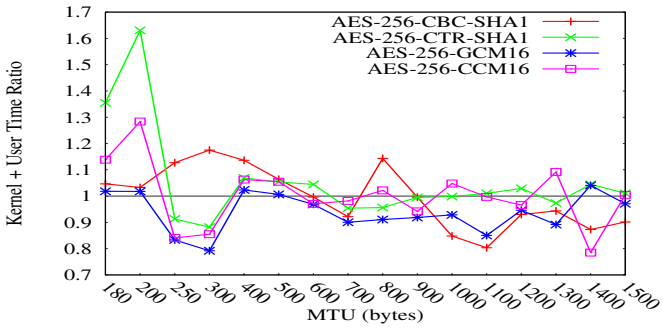
Figures 10c and 10d illustrate the ratio of the different frequency policies over the default *ondemand* policy. It shows that for huge packets, using a **governor** represents a better performance in elapsed time. This is not the case for smaller packets due to the overhead generated when changing the CPU frequency. Using a fixed frequency also reduces the time spent in the CPU, at least for frequencies higher than 1700 MHz.

*We recommend a* **governor** *fitting the required use case. Since there is no big effort in using a performance policy instead of a conservative policy, we suggest using conservative as it is better in case of power consumption. If the device has a limit number of functions, we recommend using a fixed frequency according to the function as it removes the overhead caused by the governor.*

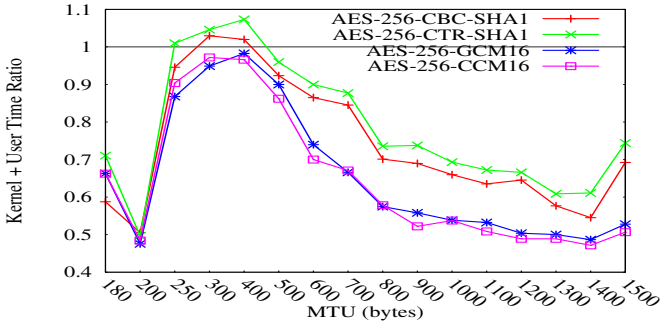
## 7. TLS VS. IPSEC




(a) *E6410* – Ratio: $\frac{4CPUs}{1CPU}$

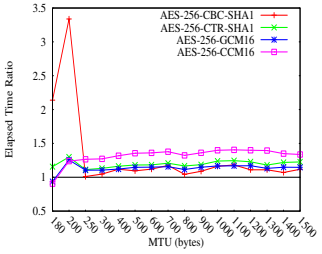



(b) *E6410* (AES-NI) – Ratio: $\frac{4CPUs}{1CPU}$

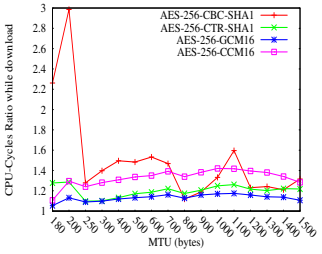



(c) *NC10* – Ratio: $\frac{2CPUs}{1CPU}$

**Figure 4: AES mode vs. multiple cores: CPU time**

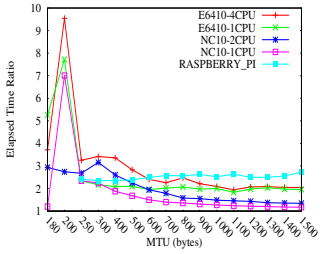



(a) Elapsed time

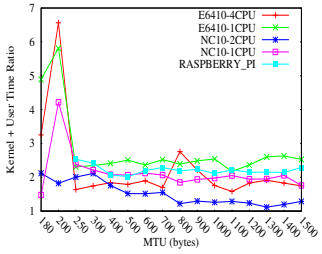



(b) CPU time

**Figure 5: Different AES modes – Ratio $\frac{sAES}{AES-NI}$**




(a) Elapsed time (transport)




(b) CPU time (transport)

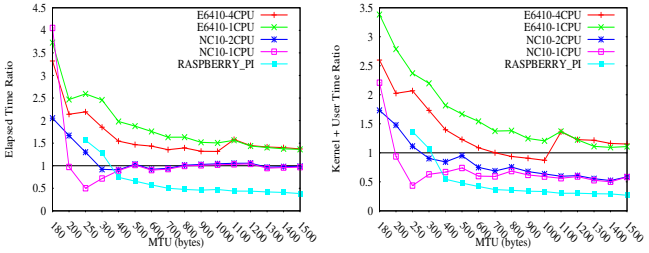
**Figure 6: Cryptographic overhead – Ratio $\frac{AES-CBC-SHA1}{NULL\ Encryption}$**

Even though there are significant design differences between TLS and IPsec, this section does not concentrate on listing them. This section actually compares end-to-end communications protected with TLS or IPsec and evaluates its performances. The major difference is that IPsec encrypts the whole TCP payload, while TLS encrypts the application payload only. For TLS, authentication (HMAC-SHA1-96) is performed once per HTTP download, whereas IPsec authenticates each IP packet and then encrypts each TCP header. Finally, the decryption process is performed within the kernel for the IPsec databases, and at the user-land for TLS.

Figure 11 compares IPsec and TLS considering huge file downloads (e.g. 500MB). Measurements over small 50Kb files fitting one single TCP frame showed similar performance.

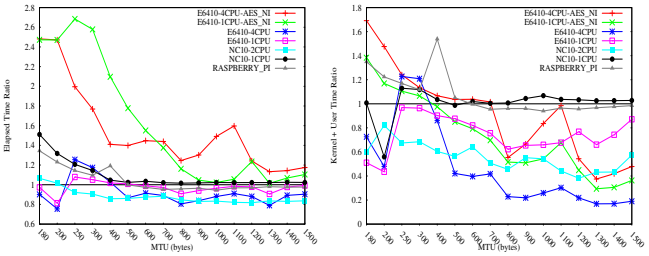
Given the additional overhead of IPsec, *RPi* and *NC10* perform poorly at least for large packets. As measured in section 5.2, the tunnel modes (optimized) outperform TLS. In addition, TLS provides huge advantage over IPsec with sAES on*E6410* , probably due to OpenSSL optimizations taking advantage of the Western architecture —stitching, PCLMULQDQ (see section 3).

*This section points out differences between IPsec and TLS. For homenet devices like RPi or NC10 , IPsec seems to be*




(a) Elapsed time (transport) (b) CPU time (transport)

**Figure 7: Network overhead – Ratio $\frac{IPsec\ Transport\ NULL\ Encryption}{HTTP\ Plaintext}$**




(a) Elapsed time (b) CPU time

**Figure 8: IPsec Tunnel overhead – Ratio $\frac{IPsec\ Tunnel\ Mode}{IPsec\ Transport\ Mode}$**

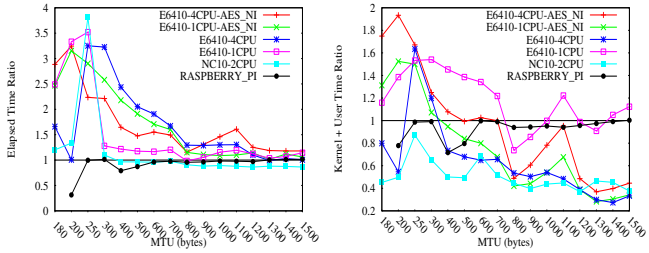
*recommended over TLS. On the other hand, TLS takes more advantage of hardware acceleration and specific instructions than IPsec does (at least for the transport mode). For sensors, IPsec and TLS are much closer as data are probably sent in a single IP packet and such devices perform all operations at the same level (no difference between kernel-land and user-land). At last, sensors do not have optimized openSSL implementation, which seems to be the main advantage of TLS over IPsec.*

# 8. CONCLUSION

There is a strong need to deploy secure communications in home networks and for Machine-to-Machine (M2M) environments. Even though most of the devices of these environments do not always benefit from the state-of-art hardware acceleration for encryption. However, it should not refrain from securing communications with authenticated encryption.

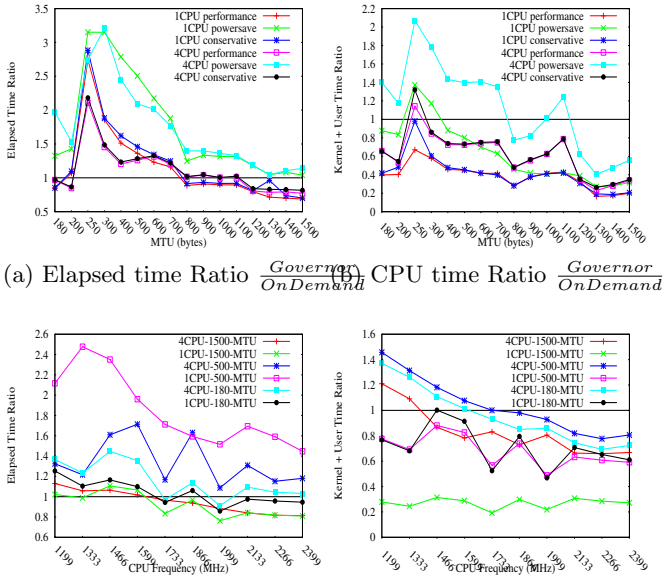
From the measurement results, our recommendations for secure home networking applications or designing home network equipments like CPE STB that perform authenticated encryption are listed below:

- **AES modes:** On a QoS aspect prefer AES-CBC and AES-CTR with HMAC-SHA1. Using a single core is sufficient and provides better QoS performance for a single communication. Hardware acceleration (AES-NI) improves significantly the QoS performance making communications $1.5 faster$, and requires 2 times less CPU cycles.




(a) Elapsed time (b) CPU time

**Figure 9: UDP encapsulation – Ratio $\frac{UDP\ Encapsulation}{IPsec\ Transport\ Mode}$**




(a) Elapsed time Ratio $\frac{Governor}{OnDemand}$ (b) CPU time Ratio $\frac{Governor}{OnDemand}$

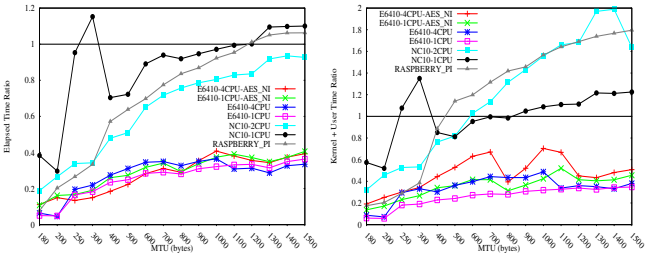
(c) Elapsed time Ratio $\frac{Frequency}{OnDemand}$ (d) CPU time Ratio $\frac{Frequency}{OnDemand}$

**Figure 10: Frequency Scaling Policies**

- **Network overhead:** Networking overhead is not negligible and is almost equivalent to the encryption overhead for large packets (without AES-NI). As tunnel increases network overhead, when possible, IPsec transport mode should be preferred and UDP encapsulation should be avoided.

- **Power usage:** For a single communication, the governor overhead may not balance its advantages. As such, for regular devices, a fix frequency may be sufficient.

- **TLS/IPsec:** TLS is likely to be more optimized for faster communications than IPsec. On the other hand, the speed of the communication is performed at the expense of more CPU cycles. As such, we would recommend IPsec for constraint devices with limited energy.




(a) Elapsed time (b) CPU time

**Figure 11: TLS vs. IPsec –Ratio $\frac{TLS}{IPsec\ Transport\ Mode}$**

## 9. ACKNOWLEDGMENTS

Testing measurements were funded and performed at Francetelecom / Orange. We would like to thank Adam Ouorou for its support. We also would like to thank Benjamin Richard for its cryptographic review and comments.

## 10. REFERENCES

[1] Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197, National Institute of Standards and Technology (NIST), Nov. 2001.

[2] ARM1176. URL: `http://www.arm.com/products/processors/classic/arm11/arm1176.php`.

[3] M. Dworkin. Recommendation for Block Cipher Modes of Operation: Methods and Techniques: The CCM Mode for Authentication and Confidentiality, 2004.

[4] M. Dworkin. Recommendation for Block Cipher Modes of Operation: Methods and Techniques: Galois/Counter Mode (GCM) for Confidentiality and Authentication, 2007.

[5] Latitude E6410 Laptop. URL: `http://www.dell.com/us/business/p/latitude-e6410/pd`.

[6] S. Farrell. Why pervasive monitoring is bad. *Internet Computing, IEEE*, 18(4):4–7, July 2014.

[7] S. Farrell and H. Tschofenig. Pervasive Monitoring Is an Attack. RFC 7258 (Best Current Practice), May 2014.

[8] V. Gopal, W. Feghali, J. Guilford, E. Ozturk, G. Wolrich, M. Dixon, M. Lochtyuhin, and M. Perminov. Fast Cryptographic Computation on IntelÂő Architecture Processors via Function Stiching. White paper, Intel, Apr. 2010.

[9] J. Granjal, J. S. Silva, E. Monteiro, and F. Boavida. Why is IPSec a viable option for wireless sensor networks. In *Mobile Ad Hoc and Sensor Systems, 2008. MASS 2008. 5th IEEE International Conference on*, pages 802–807, 2008.

[10] S. Gueron. IntelÂő Advanced Encryption Standard (AES) Instructions Set. White paper, Intel, Sept. 2012.

[11] S. Gueron and M. E. Kounavis. IntelÂő Carry-Less Multiplication Instruction and its Usage for Computing the GCM Mode. White paper, Intel, Sept. 2012.

[12] S. Gueron and V. Krasnov. The fragility of aes-gcm authentication algorithm. In *Information Technology: New Generations (ITNG), 2014 11th International Conference on*, pages 333–337, April 2014.

[13] A. Hoban. Using IntelÂő AES New Instructions and PCLMULQDQ to Significantly Improve IPSec Performance on Linux. White paper, Intel, Aug. 2010.

[14] A. Huttunen, B. Swander, V. Volpe, L. DiBurro, and M. Stenberg. UDP Encapsulation of IPsec ESP Packets, january 2005. RFC 3948.

[15] IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). *IEEE Std 802.15.4 2011 (Revision of IEEE Std 802.15.4 2006)*, pages 1–314, 2011.

[16] D. A. McGrew and J. Viega. The Security and Performance of the Galois/Counter Mode (GCM) of Operation. In *INDOCRYPT*, pages 343–355, 2004.

[17] Intel Atom Processor N270 with Mobile Intel 945GSE Express Chipset. URL: `http://ark.intel.com/de/products/36331/Intel-Atom-Processor-N270-512K-Cache-1_60-GHz-533-MHz-FSB`.

[18] 10" netbook (NC series) NP-NC10. URL: `http://www.samsung.com/ae/consumer/computers-peripherals/notebook/netbook/NP-NC10-KA01AE-spec`.

[19] Petr Svenda. Basic comparison of Modes for Authenticated-Encryption: (IAPM, XCBC, OCB, CCM, EAX, CWC, GCM, PCFB, CS), 2005.

[20] Rasberry Pi. URL: `http://www.raspberrypi.org`.

[21] S. Raza, S. Duquennoy, T. Chung, D. Yazar, T. Voigt, and U. Roedig. Securing communication in 6LoWPAN with compressed IPsec. In *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*, pages 1–8, 2011.

[22] S. Raza, D. Trabalza, and T. Voigt. 6LoWPAN Compressed DTLS for CoAP. In *Distributed Computing in Sensor Systems (DCOSS), 2012 IEEE 8th International Conference on*, pages 287–289, 2012.

[23] K. Sandlund, G. Pelletier, and L.-E. Jonsson. The RObust Header Compression (ROHC) Framework, march 2010. RFC 5795.

[24] strint report. , W3C/IAB workshop on Strengthening the Internet Against Pervasive Monitoring (STRINT), Mar. 2014.

[25] TLP. URL: `https://github.com/linrunner/TLP/blob/master/default`.