Security in Embedded Systems

Fredrik Klasson Ylva Hecktor
Email: {frekl803,ylvhe819}@student.liu.se
Supervisor: Christian Vestlund, chrve@ida.liu.se
Project Report for Information Security Course
Linköpings universitetet, Sweden

Abstract

Security in embedded systems is limited due to resource constraints. Security in embedded systems have to be adapted to these limitations. We provide a brief look at some of the limitations, such as the battery- and processor-gap. We also discuss some solutions to the limitations. In most systems the solutions are based on specialized cryptographic auxiliary processors. However there are other solutions that are more efficient but not as widely used. Two of the other solutions are the instruction- and data-driver approaches which requires hardware support. We also look at the impact of the faster growth rate of the Shannon-Hartley theorem compared to Moore's law.

1. Introduction

The goal with this report is to provide a brief insight and overview of embedded systems compared with *traditional computing* from a security point of view. We use the term traditional computing to provide the opposite of embedded systems. The definition of embedded systems is beyond the scope of this article but for this report we consider any system that is not a laptop or workstation an embedded system. For instance we consider PDAs and cellphones as embedded systems, other examples include remote controllable thermostats and any microprocessor in household appliances. Hoffmann et al. claims that a vast majority, as much as 97%, of all processors are to be considered embedded processor [2]. With this ubiquity of embedded systems security failures will potentially have a wide spread impact, and thus security is not to be neglected.

We will discuss some of the limitations of embedded systems, and the consequences of them from a security point of view. An example of a limitation is power constraints, especially for battery powered embedded systems. The limitations are what most significantly separates embedded systems from traditional computing, to understand what is special about security in embedded systems it is important to look at and understand them.

One of the reasons why embedded systems security is important is that security failures may lead to safety issues. Koopman exemplifies how a thermostat can be abused [12], this can result in anything from mere annoyance to economic loss or worse. Small children and animals may die from heat related issues. An attacker could potentially raise the temperature or turn off cooling which might lead to dangerously high temperatures.

Finally, we will address some of the limitations and what could be done to mitigate or ideally void them, but often it is as simple as a cost trade-off. This means that security that works well in traditional computing have to be adapted to the limitations of embedded systems before it is possible to implement it [10]. Many embedded systems are orders of magnitude slower or smaller than present time traditional computers - in essence they are comparable to one or a few decades old traditional computers [3].

2. Security Concerns for Embedded Systems

Gogniat et al. first divides the attacks in hardware and software [9]. The hardware attacks are further categorized as physical irreversible, physically reversible and side-channel attacks. The first two physical attacks are also referred to as active attacks whereas the side-channel attacks are passive. The reversible attacks can further be subdivided in detectable and non-detectable. Please refer to Gogniat et al. for a deeper discussion of the categorization in Figure 1. Ravi et al. makes a slightly different categorization but the general outline is the same as Gogniat et al. [5].

There might be hardware requirements such as tampering resistance [5], but also software such as buffer overflows, logical design errors and other common issues that is common for most computing. Part of the software concerns are secure communication and storage requirements. Both of which are needed in situations where the embedded systems are located in places where we do not want to transmit or store data in plain text, or if we simply want to restrict access (i.e. to ensure Confidentiality and/or Integrity of the data).

Availability is a concern that is partially both hardware and software, in the sense that batteries are hardware concerns (i.e. up-time) but software in the terms of providing access via e.g. wireless communication. An attacker can prevent availability by draining the batteries (Koopman refers to these as *battery attacks* [12]). One example of an battery attack is causing excessive network traffic that the embedded device must process. A successful battery attack could disable the embedded system even after the attack is over, a non-battery powered system probably would resume normal operation as soon as the attack is over. Identification of the user of the system can be important or superfluous, however this depends on the type of the embedded system - some are meant to serve anyone (e.g. thermostats or light switches) and some only a select group (e.g. a card/RFID reader).

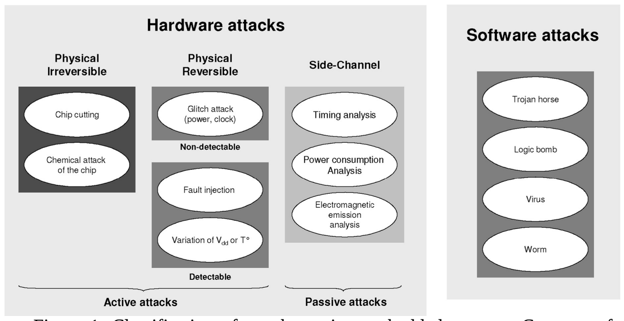

Figure 1: Classification of attacks against embedded systems. Courtesy of Gogniat et al[9]

Cost is also a concern, that might prevent the choice of bigger more expensive batteries or connecting to the power grid and laying out cables. Cost might also prevent developers from spending more time checking their code (e.g. by means of formal code reviewing) or implementing extra security features.

3. Limitations of Embedded Systems

There are several potential bottle-necks that limit security. One limitation is computational processing power, an embedded device might easily be saturated with security related computations. This is the so called *processing-gap* which we will return to in Section 3.1. The implications of the processing-gap may result in failure to maintain required operations, for instance a minimum throughput or maximum response time [5]. Real-time systems might have to use less time consuming security related computations in order to be able to schedule the tasks running on the system, depending on the real-time scheduling policy used. Embedded real-time systems are at risk of missing deadlines as a consequence of DoS (Denial of Service) attacks [12]. Embedded systems may have battery and memory constrains, *small form-factor* devices (e.g. PDAs and cellphones), are often severely constrain in this respect [5].

There are other gaps which we do not go in too much detail explaining, these are the *flexibility-*, *tamper resistance-* and *assurance-gaps* [14,15]. The flexibility-gap stresses that embedded systems have to be flexible enough to deal with different security protocols and standards. Tamper resistance-gap deals with the fact that embedded systems are facing an increasing amount of different attacks, both hardware and software attacks. Assurance-gap is related to reliability and that the system should continue to operate reliable even if it is attacked.

The *battery-gap*, is due to that batteries have a modest 5-8% capacity increase per year, whereas the energy requirements of the devices grow faster [5]. Although, in recent years, with for instance Intel's new Atom processors, the power envelope and *TDP*[1] of processors have been cut from 15W-165W for the Intel Xenon family down to 0.65W-13W for the Intel Atom series [6, 16, 17, 18].

Cost is mentioned as one aspect of limitations [5,12], however the cost is also a limitation of traditional computing of similar impact. However according to Koopman embedded systems are very cost sensitive [12]. In essence the more money one has the more resources can be put on mitigating other factors (e.g. development time, code-review time, more expensive physical tamper-protection or detection). One impact of the battery-gap limitation is that it directly translates to the availability, i.e. up-time, of the device. Since most communication, and wireless communication in particular, requires power proportional to the amount of data being transmitted, as a consequence this imposes limitations in the amount or rate of data transmissions.

3.1 An Example of the Processing-gap

As mentioned above, part of the processing-gap can be attributed to the battery-gap. Much of the processing-gap stems from the impact of *secure computations* and communication overhead, to illustrate the impact we will borrow an example from Ravi et al., that takes a handheld iPAQ H3670 PDA using SSL to secure communication. The iPAQ H3670 contains an Intel SA-1110 StrongARM processor clocked at 206 MHz [5]. Their research found that if 10% of the resources of the SA-1110 processor was used the result would be that a data rate less than 180 kbps was achieved [5]. Many embedded systems, such as cellphones, use wireless communication technologies such as GSM EDGE (Enhanced Data rates for GSM Evolution [8]) or IMT-2000 (a.k.a. “3G”). We will look at the bandwidth of those communication technologies and compare it to the data rate of 180 kbps.

An Ericsson AB white paper from September 2009 that outlines the evolution of EDGE claims that *user bit rates* of up to 250 kbps are available, this would leave 28% of the bandwidth unused for the Intel SA-1110 example [8]. Furuskär et al. summarizes the IMT-2000's data rate requirements to between 384 kbps and 2 Mbps, depending on the coverage requirements [7]. Calculating with a lower bound of 384 kpbs gives 53% unused

[1]Thermal design power, which is an estimation of the maximum power drain during normal usage, actual usage might be both more or less

bandwidth for the Intel SA-1110 example. The data rates and processor speeds are merely snapshots of a point in time, thus it is from a research point-of-view of more interest to look at the asymptotic behavior to see if this is just a temporary gap or if the bandwidth waste will grow. Note that we are now disregarding from the resources needed to perform encryption and decryption, and only looking at the cost of channel coding[2]. Even though specialized network hardware usually handles the channel coding/decoding those parts are also embedded processors themselves and part of the whole larger embedded system, sharing the same battery and other resources. Ravi et al. gives an example of what they call "*high-end embedded system*" (exemplified with VPN servers and firewalls) using an Intel Xenon processor clocked at 2.8 GHz. If 100% of the Xenon processor was used they were only able to get a data rate of less than 29 Mbps[5]. For embedded systems like VPN routers 29 Mbps might be unacceptable low. Consider a wired 100 Mbps network, this would be 71% unused bandwidth. With network cards capable of 1 Gbps speeds not being uncommon on newer (traditional) computers the unused bandwidth will be even greater. Hoffmann et al. explains that "Shannon asks for more than Moore can deliver!" [2]. "Shannon" is a reference to the Shannon-Hartley theorem (a.k.a "Shannon's law"), which is providing a growth rate for algorithmic complexity of channel coding. In essence it tells us the theoretical upper bound of the maximum speed at which we can reliably transmit data on a noisy channel. The growth rates are visualized in Figure 2. "Moore" referees to Moore's law that provides an estimate for the expected growth of processors' transistor count over time (which is generally translated to expected performance increase). In other words Hoffmann et al. thus claims that the unused bandwidth will keep growing as time goes on (in the long term, we would even be unable to fully utilize the channels full bandwidth regardless of if we use encryption or not) [2]. Further more in recent years the raw processor speeds of traditional computers has stagnated between 1-3 GHz per core.

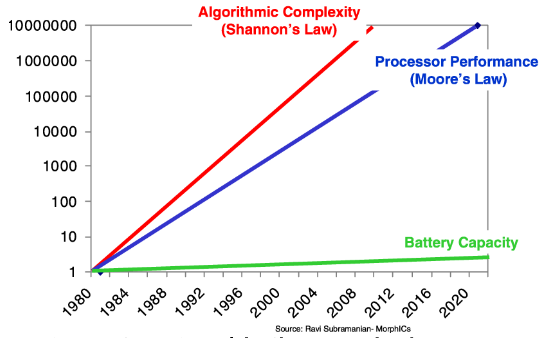

Figure 2: Comparison of the Shannon-Hartley theorem, Moore's law and battery capacity growth. Image courtesy of Ravi Srivaths.

There has been a shift towards many-core, the impact of which is yet to be seen (for instance this far only two of Intel's 19 Atom processors are dual core, the D510 and the 330 models [11]). It is worth pointing out that in the long term the implications of Shannon-Hartley theorem will also affect traditional computers' ability to utilize full bandwidth regardless of encryption. However embedded systems are comparable to old traditional computers with respect to their limitations such as processing capabilities. Due to this fact they are as illustrated by the SA-1110 example above already seeing the effects of the Shannon-Hartley theorem [3]. The processing-gap means that from a security perspective we might not be able to use as much encryption as we would like when we also seek high throughput. This is a concern mainly for embedded systems expected to have a high throughput while providing strong security such as VPN routers or firewalls. These devices also need to devote processing power to perform other tasks than just pure encryption/decryption (e.g. packet filtering logic, routing decisions), making the situation even worse by adding more computational requirements.

4. Security Solutions to Limitations in Embedded Systems

Security in embedded systems puts an extra strain on the limited embedded systems. Due to this fact the security solutions of embedded systems have to be adapted to systems they are used for [13]. There are two different approaches to address the problems with the processing-gap. The first one is to reduce the security processing workload, the other way is to enhance the processing capabilities of the embedded system [5].

The development of crypto systems has lead to new more efficient algorithms, such as ECC, AES and NTRU [19,20,21]. However these are not widely used due to a preference for conventional and well known algorithms [5]. Although the newer ones are faster, the security risks are unknown. For example flaws have been found in the original NTRU signature scheme, as well as in the updated version that should have solved the problems [5].

4.1 Hardware security Solutions

There have been several attempts to increase the security processing capabilities of general purpose processors. Due to the fact that most processors today are word oriented, the main target have been to accelerate the bit-level arithmetic operations [5]. Different instruction set extensions have been proposed. Many such extensions to processors have already been made to embedded processors in wireless handset devices [5].

Other hardware solutions that might be used to enhance the security are *cryptographic processors. The* specialized cryptographic processors offloads the cryptographic computations from the main processor to the cryptographic hardware, easing the main processor's workload. The cryptographic hardware can be designed for high performance and low power consumption [15].

[2]Encoding the data for transmission over the communication channel.

Most of the security solutions for the limitations in embedded systems are based on cryptographic processor solutions. However other solutions to the limitations are important, since the cost of processor based solutions are high [15]. A development of the hardware solution is to let the processor offload large portions of the security protocol, and not just the cryptographic algorithm, to a *security protocol processing engine* [5]. Security processing engines accelerate most of the security protocols functionality, and these engines are programmable and therefore can be used for multiple protocols.

One way of implementing security is with *reconfigurable architecture* and hardware monitors. The reconfigurable architecture enables the implementation of security primitives. A security primitive corresponds to a hardware accelerator and performs a security algorithm. The device performs multiple security primitives independently of each other. Different primitives are used to fulfill different security goals. The goals could be speeding up the computations of the security algorithms, in comparison to the speed of pure software solutions. Another goal is the flexibility to update or switch to another security primitive as needed. A fixed hardware implementation lacks this flexibility. Reconfigurable architecture meet real-time constraints, such as the power and reliability constraints [15]. Trough monitoring, abnormal behaviors in the system can be detected and attacks can be avoided.

4.2 Instruction- and Data-driven Approaches

Another way to reduce the strain that security puts on the limited embedded system is to limit the security solutions to the parts of code/data that need to be protected. There are two different ways of deciding which parts of the code that are necessary to protect [13]. The first is based on *instruction-driven security*, this static approach analyzes the program at compile time to find all the instructions that have operands that operate on variables that should be protected. The second is based on *data-driven security* and is a dynamic approach. At run-time it analyzes what instructions and data to protect. When the processor is executing an instruction that has an operand that is marked as secure it treats the instruction as secure.

4.3 Power Profile Analysis

Power profile analysis is when an attacker use the pattern of the power consumption to get information about the keys used in the system. The power consumption is correlated to the operation the hardware device is performing [5]. One way to prevent or at least make it harder to perform power profile analysis is to mask the power profile of instructions, this can done by for instance computing complementary bits. Masking the power consumption increases the overall power consumption of the processor, something which puts more strain on the batteries on battery powered systems. Saputra et al. suggests a data-driven approach that requires changes to the hardware architecture [13]. The ARM TrustZone is one example of such an architecture where Saputra's et al. data-driven solution could be implemented. By extending the implementation to select power masking variants of the instructions when operating on data tagged with the security tag called the S-bit. The S-bit is used to create division between the two zones of the ARM TrustZone architecture. According to Ravi et al. the basic primary goal of *TrustZone* is the creation of a separated secure kernel separating trusted and untrusted code [5,13]. In essence also using the S-bit for power masking would be using the instruction driven approach since not necessarily all operations in the trusted zone operates on data needing security.

5. Discussion

We think that the most severe limitations of embedded systems security is the battery- and processor-gap which are rather tightly coupled. Since faster processors generally require more power, although Intel's new Atom series has reduced the power requirements while still keeping processor speeds above the 1 GHz line. In our opinion the processor-gap still is the most severe since it is likely to continue to be a limitation due to the higher growth of complexity in communication compared to processor speed.

The implications of the processing-gap is that developers are forced to choose less security in order to provide high performance. Another issue is that the cost of the embedded systems have to be increased to allow for use of more costly components. Larger batteries or specialized auxiliary processors that offload the main CPU (e.g. FPGAs which could be programmed to more efficiently perform security related computations). We have seen that to deal with security in embedded systems flexibility is becoming important. Therefore the more general programmable processors will probably be coming more into use instead of hard wired cryptographic processors.

Other solutions require more drastic changes such as new processor techniques, for instance the ARM TrustZone which requires changes in the design of not just the processor but also the memory to accommodate security information. As mentioned earlier, the ARM TrustZone could be extended to allow for power masking variants of the instructions when operating on secure data, at the expense of higher power consumption. Especially for the naïve approach of power masking everything executing in the trusted zone with the S-bit. One could alternatively choose the static approach and put the power masking tag in the instruction code words. The work of Saputra et al. [13] suggests that the dynamic, data-driven approach is the most efficient, since it gives opportunity to use more energy efficient instructions when operating on data not tagged as secure. Thus only employing protection on operations working on secure data or the result of other operations, where at least part of the input was protected. It is worth pointing out that TrustZone and the data driven approach of Saputra et al. [13] have been designed with different intent, separation of *control* depending on (dis-)trust compared to selective *data* protection depending on the data.

Following the discussion above, it becomes fairly obvious that a naïve merge, where all code in the TrustZone use power masking instruction variants, would be comparable to the static approach of Saputra et al. [13]. Since not all trusted code necessarily operate on data that needs protection. Saputra et al. [13] demonstrates that on average the static approach is 16% less efficient than the dynamic approach. It might be beneficial to combine the TrustZone with the ideas of Saputra et al. but this requires more research.

6. Conclusions

The most severe limitations are the processing- and battery gaps. The processor-gap is mainly due to the higher growth of the communication complexity compared to the evolution of processors. Various mitigation strategies have been proposed. However these mere postpone the need to lessen either the requirements on security or performance of the embedded systems. Solutions may require changes to the architecture of the embedded systems. Changes may be to add auxiliary processors to the embedded system or more fundamental changes including memory layout, such as the ARM TrustZone.

References

[1] Madhukar Anand and Insup Lee. 2008. Challenges and opportunities in deeply embedded systems security. *SIGBED Rev. 5, 1*, Article 25 (January 2008), 2 pages.

[2] *N.B: Secondary source in Zambreno et al. [4] references:* Andreas Hoffmann, Heinrich Meyr, and Rainer Leupers. 2002. *Architecture Exploration for Embedded Processors with Lisa*. Kluwer Academic Publishers, Norwell, MA, USA.

[3] Arbaugh, W.A.; van Doorn, L.; , "Embedded security: challenges and concerns," *Computer* , vol.34, no.10, pp.40-41, Oct 2001

[4] Joseph Zambreno , Alok Choudhary , Rahul Simha , Bhagi Narahari , Nasir Memon, SAFE-OPS: An approach to embedded software security, *ACM Transactions on Embedded Computing Systems (TECS)*, v.4 n.1, p.189-210, February 2005

[5] Srivaths Ravi, Anand Raghunathan, Paul Kocher, and Sunil Hattangady. 2004. Security in embedded systems: Design challenges. *ACM Trans. Embed. Comput. Syst.* 3, 3 (August 2004), 461-491.

[6] "Intel® Xeon® Processor ULV 1.66 GHz, 2M Cache, 667 MHz FSB with SPEC Code(s):",Intel Inc., *Intel product specifications*, http://ark.intel.com/Product.aspx?id=29750 (accessed at 2010-04-28)

[7] Furuskär, A.; Mazur, S.; Müller, F.; Olofsson, H.; , "EDGE: enhanced data rates for GSM and TDMA/136 evolution," *Personal Communications, IEEE , vol.6, no.3*, pp.56-66, Jun 1999

[8] "The Evolution of Edgde", Ericsson AB, Ericsson *White Papers*, Sep 2009. Available at http://www.ericsson.com/article/evolution_of_edge_20100210105126 (Last accessed 2010-04-16)

[9] Gogniat, G.; Wolf, T.; Burleson, W.; Diguet, J.-P.; Bossuet, L.; Vaslin, R.; , "Reconfigurable Hardware for High-Security/ High-Performance Embedded Systems: The SAFES Perspective," Very Large Scale Integration (VLSI) Systems, *IEEE Transactions on , vol.16, no.2*, pp.144-155, Feb. 2008

[10] Vaslin, R.; Gogniat, G.; Diguet, J.-P.; Pegatoquet, A.; , "Trusted computing - A new challenge for embedded systems," *Electronics, Circuits and Systems, 2006. ICECS '06. 13th IEEE International Conference on* , vol., no., pp.776-779, 10-13 Dec. 2006

[11] "Intel® Atom™ Processor Specifications". http://www.intel.com/products/processor/atom/specifications.htm. Accessed at 2010-04-19.

[12] Koopman, P.; , "Embedded system security," *Computer*, vol.37, no.7, pp. 95- 97, July 2004

[13] Saputra, H.; Ozturk, O.; Vijaykrishnan, N.; Kandemir, M.; Brooks, R.; , "A data-driven approach for embedded security," *VLSI, 2005. Proceedings. IEEE Computer Society Annual Symposium on*, vol., no., pp. 104- 109, 11-12 May 2005

[14] Gogniat, G.; Wolf, T.; Burleson, W.; , "Reconfigurable Security Primitive for Embedded Systems," *System-on-Chip, 2005. Proceedings. 2005 International Symposium on*, vol., no., pp.23-28, 17-17 Nov. 2005

[15] Gogniat, G.; Wolf, T.; Burleson, W.; , "Reconfigurable Security Support for Embedded Systems," System Sciences, 2006. HICSS '06. Proceedings of the 39th Annual Hawaii International Conference on, vol.10, no., pp. 250a- 250a, 04-07 Jan. 2006

[16] "Intel® Xeon® Processor 7020 (2M Cache, 2.66 GHz, 667 MHz FSB) with SPEC Code(s) SL8UA:",Intel Inc., *Intel product specifications*, http://ark.intel.com/Product.aspx?id=27224 (accessed at 2010-04-28)

[17] "Intel® Atom™ Processor Z500 (512K Cache, 800 MHz, 400 MHz FSB) with SPEC Code(s) SLB6Q:",Intel Inc., *Intel product specifications*, http://ark.intel.com/Product.aspx?id=35472 (accessed at 2010-04-28)

[18] "Intel® Atom™ Processor D510 (1M Cache, 1.66 GHz) with SPEC Code(s) SLBLA:",Intel Inc., *Intel product specifications*,

http://ark.intel.com/Product.aspx?id=43098 (accessed at 2010-04-28)

[19] Menezes, A. J. 1994 *Elliptic Curve Public Key Cryptosystems*. Kluwer, Academic Publishers.

[20] AES *Algorithm (Rijndael) Information*. Available at http://csrc.nist.gov/encryption/aes/rijndael

[21] NTRU *Communications and Content Security*. Available at http://www.ntru.com