센서 네트워크를 위한 부가적인 암호모듈의 구조 분석

김정태
목원대학교

Analyses of additive Crypto-module Architecture for a Sensor Network

Jung-Tae Kim
Mokwon University
E-mail : jtkim3050@mokwon.ac.kr

요 약

In this paper, we analyses of additive crypto-module architecture for a sensor network. Recent research in sensor networks has raised security issues for small embedded devices. Security concerns are motivated by the development of a large number of sensor devices in the field. Limitations in processing power, battery life, communication bandwidth and memoryconstrain devices. A mismatch between wide arithmetic for security and embedded data buscombined with lack of certain operations. Then, we compared the architecture of crypto-module in this paper.

I. Introduction

Sensor networks offer economically viable solutions fo a variety of applications. Networked microsensors technology is a key technology for the future. Cheap, smart devices with multiple onboard sensors, networked through wireless links and Internet and deployed in large numbers, provide unprecedented opportunities for instrumenting and controlling homes and the environment. In addition, networked microsensors provide the technology for a broad spectrum of systems in the defense arena, generating new capabilities for reconnnaissance and surveillance as · well as other tactical applications. Recent advances in computing and communication have caused a significant shift in sensor network research and brought it closer to achieving the original vision. Small and inexpensive sensors based upon microelctromechanical system technology, wireless networking, and inexpensive low-power processors allow the deployment of wireless ad hoc networks for various applications. Again, DARPA stated a research program on sensor networks to leverage the latest technological advances. The recently concluded DARPA sensor information technology program pursued two key research and development thrusts. First, it developed new networking techniques. In the battlefield context, these sensor devices or nodes should be ready for deployment, in an ad hoc fashion, and in highly dynamic environments. Recent work in sensor networks allow the collection of data from low-end sensor nodes in the field. This data is communicated over non-secure channels, such as radio frequencies, though routers and, ultimately, to a base station for further processing and decision making. Applications range from battlefield surveillance over data collection to study environmental impacts to medical observation. Beyond sensor networks, embedded processors are *increasingly deployed with network* connections, such as in PDAs with wireless communication.

Ⅱ. Algorithms for Encryption

Our choice of algorithms represents popular symmetric encryption and hashing function schemes that form an integer part of many security protocols. RC4 is used in IEEE802.11 WEP, IDEA and MD5 are part of PGP, SHA-1 and MD5 are included in the security architecture for Internat protocol. These algorithms offer variety in the mode in which they operate and encompass different mathmatical and data manipulation operations. They work on different word sizes ranging from 8 bits to 32 bits, and hence, help assess the effectiveness of the different architecture. Table 1 presents the parameter in analyses.

Table 1: Encryption Schemes and Parameters

Algorithm	Type	key/hash (bits)	Block (bits)
RC4	stream	128	8
IDEA	block	128	64
RC5	block	64	64
MD5	1-way hash	128	512
SHA-1	1-way hash	128	512

1) RC4 : stream cipher symmetic key algorithm. This algorithm is quite simple and operations involve the addition of 8 bit elements or swapping variables in a 256 byte state table. RC4 supports variable length keys. We consider a 128 bit key here.

2) IDEA : symmetric key block cipher that *operates on 64 bit plaintext blocks.* The key is 128 bits cipher that operates on 64 bit plaintext blocks. The key is 128 bits long with the same algorithm used for both encryption and decryption. The algorithm primarily includes operations from threealgebraic group: XOR, addition modular 216, multiplication modulo 216+1

3) RC5 : a fast symmetric block cipher with a variety of parameters block sixe, key size and number of rounds. We currently focus o a RC5 implementation with a 64 bit data block and 64 bit key. It uses the XOR, addition and rotation operations.

4) MD5 : one-way hash function that processes the input textin 512 bit blocks to generate a 128 bit hash. The mathmatical operations that are involved in this algorithm are: XOR, AND, OR, NOT and rotations. The algorithms also pads plaintext to 512 blocks with the last 64 bits of the last block indicating the length of the message

5) SHA-1 : also one way hash function that produces a 160 bit output when any message of any length less than 264 bits is input. The operations are similar to MD5 and constitute XOR, AND, ORM NOT and rotations

Ⅲ. Performance Model

We observed that the word length and architectural features, namely the complexity of the ISA and supports for certain ALU operations are the causes of variations. From these findings and the experimental data, we can derive a multi-variant model that allows the interpolation of performance for other architecture. The objectives of such a model are threefold. First, feasibility od existing encryption schemes can be derived by just implementing one scheme on an architecture. Second, encryption overhead can be assessed based on architectural parameters to drive architecture design for a specific encryption scheme and formulate minimum requirements. Third, new encryption schemes only need to be assessed on a subset of reference platforms while their performance on other platforms can be derived from the method. First, a simple model is introduced. The results of this model is imprecise as there are many variables that influence the execution times of any program. The objectives of this model is to aid a designerin computing a rough estimate of the execution times for a givenencryption algorithm and a particular microprocessor. We derived the following performance model:

$$t_{exec}(txt\_len) = \frac{a + b\left(text\_length\ \text{h/blocksiz e}\right)}{\text{processor}\quad \text{freq} * \text{bus\_width}}$$

where () is the ceiling function, text_length is the size of the plaintext in bytes, processor_frequency and bus_width are the frequency and bus width of the microcontroller, respectively. The parameters a snd b depend on the algorithm beign evaluated, and block_size is the sixe of the blocks in the algorithm. Parameter a includes all the initialization overheads while b aptures time spent in operatins repeated for each block.

The model in equation is refined to account for other parameters that affect the execution times, For example, some algorithms can take advantage of the existence of a multiply instruction. A more detailed model for the parameters a snd b can be derived as follows.

Table 2. Parameters for performance model

Algorithm	A	B	Block size(bits)
MD5	203656	86298	512
SHA-1	60980	458660	512
RC5/encrypt	352114	40061	64
RC5/decrypt	352114	39981	64
IDEA encrypt	67751	80617	64
IDEA decrypt	385562	84066	64
RC4	68540	13591	8

$$a = a_{BASE} + a_{MUL} + a_{RISC}$$
$$b = b_{BASE} + b_{MUL} + b_{RISC}$$

where a_{BASE} and b_{BASE} arethe base parameters shown in Table 2, a_{MUL} and b_{MUL} are adjustments of those parameters, which take into account the presence of absence of a multiplication instructions and a_{RISC} and b_{RISC} take into account the type of the microprocessor architecture.

Ⅳ. Simulation Result

Table 2 depicts the excution time overhead for each of the considered platforms and algorithms on a log scale. For the digest algorithms, we used multiple plaintext sizes to empasize the non-linear behavior of those algorithms with the length of the plaintext. The main reason for this nonlinear behaviors is the existence of a minimum plaintext sizefor those algorithms, so smaller messages are padded up to the minimum plaintextsize. As expected, the slowest microcontroller, which is also the simplest, will take longest time to complete any of the analyzed cryptography algorithms. The results is presented in Table 3.

Table 3. Execution time for algorithms

Algorithm	size	Atmega 103	Strong Arm	Xscale
MD5	0	5863	46	26
	1-26	5890	46	26
	62-8	10888	74	45
SHA-1	1	15249	69	51
	56	14543	133	102
	64	31107	145	103
RC5	16	9641	41	45
IDEA	16	1523	26	21
RC4	16	1886	155	108

V. Conclusion

In this paper, we presented a survey investigating the computational requirements for a number of cryptographic algorithms and embedded architecture. We also derived a model to assess the computational overhead of embedded architecture for encryption protocols in general. Our analytial model assesses the impact of arbitary embedded architectures as a multi-variant function for each encryption schemedepending on processor frequency. Our scheme are not only valuable to assess the feasibility of encryption schemes for arbitrary embedded architectures, but also provide the basis for modeling encrypttion overheads across platforms.

References

[1] D. Wheeler, M. Needham, "TEA, a Tiny Encryption Algorithm", Fast Soft Encryption: Second International Workshop", Springer LNCS, vol 1008, 1994 pp.14-16

[2] C. Schnorr, Efficient signature generation by smart carts, Journal of Cryptology, vol.4, page 161-174, 1991

[3] A. Perrig, R. Szewczyk, "Security Protocols for Sensor Networks", Wireless Networks, vol.8, no.5, pp. 521-534, 2002

[4] W. Fumy and P. L풍개차, "Principles of key management", IEEE Journal of Selected Areas in Communications, vol.11, pp.785-793, June 1993.