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 memory constrain devices. A mismatch between wide arithmetic for security and embedded data bus combined with lack of certain operations. Then, we compared the architecture of crypto-module in this paper.

I. Introduction

Sensor networks offer economically viable solutions to 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 reconnaissance 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 microelectromechanical 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 Internet
protocol. These algorithms offer variety
in the mode in which they operate and
encompass different mathematical 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>
<thead>
<tr>
<th>Algorithm</th>
<th>Type</th>
<th>key/hash (bits)</th>
<th>Block (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC4</td>
<td>stream</td>
<td>128</td>
<td>8</td>
</tr>
<tr>
<td>IDEA</td>
<td>block</td>
<td>128</td>
<td>64</td>
</tr>
<tr>
<td>RC5</td>
<td>block</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>MD5</td>
<td>1-way</td>
<td>128</td>
<td>512</td>
</tr>
<tr>
<td>SHA-1</td>
<td>1-way</td>
<td>128</td>
<td>512</td>
</tr>
</tbody>
</table>

1) RC4 : stream cipher symmetric 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 three algebraic 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 on 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 text in 512 bit blocks to
generate a 128 bit hash. The mathematical
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 of 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 designer in computing a rough estimate of the execution times for a given encryption algorithm and a particular microprocessor. We derived the following performance model:

\[ t_{enc}(\text{text\_len}) = \frac{a + b(\text{text\_length} \cdot \text{blocksiz}e)}{\text{processor\_freq} \cdot \text{bus\_width}} \]

where \( \lceil \cdot \rceil \) 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 \( a \) and \( b \) depend on the algorithm being evaluated, and block\_size is the size of the blocks in the algorithm. Parameter \( a \) includes all the initialization overheads while \( b \) captures time spent in operations 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 \( a \) and \( b \) can be derived as follows.

**Table 2. Parameters for performance model**

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

\[ a = a_{\text{BASE}} + a_{\text{MUL}} + a_{\text{RISC}} \]

\[ b = b_{\text{BASE}} + b_{\text{MUL}} + b_{\text{RISC}} \]

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

**IV. Simulation Result**

Table 2 depicts the execution time overhead for each of the considered platforms and algorithms on a log scale. For the digest algorithms, we used multiple plaintext sizes to emphasize 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 size for those algorithms, so smaller messages are padded up to the minimum plaintext size. 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

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

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 analytical model assesses the impact of arbitrary embedded architectures as a multi-variant function for each encryption scheme depending 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 encryption overhead across platforms.

References