ABSTRACT
Wireless embedded systems, especially life-critical body-area networks, need security in order to prevent unauthorized and malicious users from injecting traffic and accessing confidential data. Coupled with the security costs in system performance and power consumption, embedded systems are also restricted by the type of security that can fit in their limited memory. To address these issues, we introduce a Dynamic Security System (DYNASEC), a novel architecture that ensures message integrity and confidentiality in wireless embedded systems. A delay-aware heuristic adapts the security levels and the amount of sensor data transmitted in a sensor network to meet timing constraints. This experimental analysis on a reconfigurable electrocardiogram (ECG) application validates the efficacy of the DYNASEC architecture in a body area network. Our experiments demonstrate that DYNASEC enables lightweight medical embedded systems to dynamically optimize security levels to meet timing constraints in a body sensor network.

Keywords
Performance evaluation, security, embedded systems, body area networks, medical applications, adaptable systems.

1. INTRODUCTION
Security is becoming increasingly important in wireless embedded systems, especially when the data being transmitted is life-critical and/or confidential via legal mandate. This is most important in body-area networks where motes embedded on the patient’s body may monitor a potentially fatal health condition. Security is challenging, however, because embedded systems typically have limited battery supplies and processing power coupled with the fact that real-time applications have stringent timing constraints that cannot be violated. Within the security community, the trend has been toward stronger cryptographic algorithms with increased processing requirements; meanwhile, increased wireless communication rates have placed further strain on battery lifetimes. The result is a security-processing gap, a term which recognizes the collective disparity between security requirements and the processing capabilities of embedded processors [13]. Wireless communication in networked embedded systems, furthermore, is limited by bandwidth and power. Such systems must minimize the number of packets sent in order to conserve precious power resources.

Wireless networks are more vulnerable to security attacks than wired networks, because the malicious intruder simply needs to activate an antenna to eavesdrop or inject data into a network rather than physically compromising a wire. Consequently, data integrity and confidentiality must be ensured while maintaining network availability; however, approaches to security in embedded systems must deviate from those employed for wireless networks due to limited capabilities of the nodes in the network.

As an example, consider a wireless network in the healthcare industry. The Health Insurance Portability and Accountability Act (HIPAA) mandates that healthcare professionals “ensure the integrity and confidentiality of the information” and “protect against any reasonably anticipated threats or hazards to the security or integrity of the information” [8]. If an abnormal health-related activity is detected, then additional processing power is required and extra packets must be transmitted to take care of the situation within hard real-time constraints. The cost of security under normal operating conditions may be too high to allow the deadline to be met. To address this issue, we propose a reconfigurable approach that lowers the strength of the security while enabling the system to reach its deadline; however, the security should be lowered no further than the absolute minimum that allows the system to meet its deadline.

To meet these needs, we introduce the Dynamic Security System (DYNASEC), a reconfigurable security architecture for the Sensor Operating System (SOS). SOS provides code migration, a service that enables one embedded system to download a program or module to execute on another [5]. Unfortunately, this makes SOS particularly vulnerable to attacks on authentication, whereby an attacker injects a malicious program onto a node which will then execute it. This type of attack can be countered by using symmetric key cryptography that authenticates each message with a MIC (Message Integrity Code), a secure checksum of the message. Snooping is prevented by encrypting the messages with either the Skipjack or RC5 ciphers. DYNASEC varies the strength of the cryptographic algorithms used by SOS in order to ensure that real-time constraints are met.

The primary contributions of this paper include:

(1) A lightweight delay-aware heuristic that selects the maximum level of security while meeting timing constraints.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. BodyNets’07, June 11-13, 2007, Florence, Italy. Copyright 2007 ICST 978-963-06-2193-9.
(2) An experimental analysis in a simulation that establishes DYNASEC’s ability to maximize security while meeting timing constraints. The overall goal of DYNASEC is to dynamically maximize security settings in order to meet timing constraints. This sets DYNASEC apart from other established security protocols for embedded networks that have not addressed timing constraints in-depth [9][4][7]. DYNASEC’s reconfigurable security architecture has two modes: (1) integrity and (2) integrity+encryption. Integrity verifies that messages were not modified in transit; encryption ensures that only authorized nodes can read the information, which is broadcast across an otherwise insecure wireless channel. DYNASEC also implements two lightweight cryptographic protocols: Skipjack and RC5.

2. RELATED WORK
DYNASEC meets the security and power utilization needs of the next generation of networked embedded systems. One of the most important design challenges for such systems is flexibility [13]. DYNASEC allows an embedded system sufficient flexibility to adapt to changing system requirements and to reprogram the embedded device via wireless code migration. Prior research has focused on implementing optimal security protocols on embedded systems or analyzing power consumption on embedded systems [3][17][9][4][7][12]. Some papers have also mentioned flexibility, reconfigurability, and adaptive execution of security protocols as future work [15][12] but have not tackled the problem directly.

Modifying data resolution was explored in [1] where the granularity and number of sensors were modified using a cost based decision tree. [6] also modified data granularity using Bayesian classifiers to infer context. Our work differs by presenting an algorithm that analyzes how the network, not only the embedded system, reconfigures to meet deadlines using different security levels and data sizes.

TinyPK [17] is a security scheme that provides authentication and key exchange using RSA encryption for embedded systems. Malan et al. [9] describe the first known implementation of elliptic curve cryptography (ECC) for embedded systems on the mica2 mote and Gura et al. [4] describe an implementation of RSA and ECC on mica2 motes using optimized assembly code. TinySec [7] is a link layer security scheme for wireless sensor networks using authentication and Skipjack and RC5 encryption. DYNASEC provides similar functionality as TinySec in that it encrypts data bit by bit as it is transported over the radio. In contrast to all the above security systems, DYNASEC changes the cryptographic algorithms depending on timing constraints. In DYNASEC, if timing constraints do not permit the encryption to be used, it can be swapped in favor of a lighter-weight protocol.

3. DYNASEC SYSTEM DESIGN
DYNASEC offers link-by-link integrity and encryption that has been implemented in a SOS [5]. SOS uses a message passing system to sever ties between the core OS and individual applications (modules). SOS can load or remove modules at runtime without interrupting the core OS, and the modules support autonomous message handling. To ensure integrity, DYNASEC computes a Message Integrity Code (MIC) over the header and payload of the packet. The payload can also be encrypted after the MIC has been computed.

DYNASEC was implemented in the SOS kernel. Kernel space was chosen instead of user space because it is more secure against malicious code. Unfortunately, this increased the size of the kernel, from 17% of internal memory originally to 39%. The reason for the increased kernel size in DYNASEC is the tables (e.g. s-boxes) required for each cryptographic algorithm; the code size is not particularly large. This highlights the need for new cryptographic algorithms for memory-constrained devices that specifically attempt to minimize the size of the data segment; this issue, however, is beyond the scope of DYNASEC.

DYNASEC uses cipher block chaining (CBC-MIC), for computing and verifying MICs. CBC-MIC is efficient and fast, relies on a block cipher, and minimizes the number of cryptographic primitives implemented in memory. CBC-MIC is secure, but the messages must be a standard size. DYNASEC currently supports the Skipjack [2] and RC5 [14] ciphers; the implementations are based on those described by [7]. RC5 is a stronger cryptographic algorithm than Skipjack, and it also runs faster. However, RC5 requires that a pre-computed key schedule be stored in memory taking up 104 bytes for each key, 2.6% of the total RAM in a Mica2 mote. RC5 is patented, however, making it less appealing for open-source and academic projects.

An important design decision was to keep the original message structure allowing for backwards compatibility with pre-existing SOS applications. As a message passing system, SOS uses the same message structure for all kinds of communication: (1) between different nodes using the network and (2) between different modules within the same node. The new DYNASEC fields were incorporated in the original SOS message format; however, additional changes were required to the radio driver, since the new fields transmitted over the radio are dependent on the security mode used by the application.

Data is encrypted as it was sent over the radio bit by bit. The advantage of encrypting the data over the radio is to avoid saving two large data structures for the encrypted and decrypted data in memory. This imposes the constraint that the data cannot spend more time being encrypted/decrypted than it takes to send a byte. If only authentication is set, then the IV is not transmitted; the MIC is computed over the packet, headers and payload. If encryption is set, then the IV and the MIC are transmitted, and only the payload is encrypted. When receiving a packet, the code waits for the reception of the flag byte. When this byte is received, the appropriate bits are checked and, depending on the security modes used to send the packet, the MIC/de-encryption routines may be called.

The Mica2 radio driver was modified for to accommodate the different security modes. The Mica2 radio driver in SOS is a state machine, with different submachines for transmission and reception. New states were added to each submachine to handle authentication and encryption in the radio.

4. MEDICAL APPLICATION
DYNASEC is designed for resource-constrained devices. Body Area Networks (BANs) are typically comprised of lightweight medical systems with embedded sensors to monitor physiological activities of the body or to probe the outside environment. Ubiquitous BANs may contain non-invasive and in-vivo sensors. We focus specifically on the electrocardiogram (ECG) sensor.
A distinctive characteristic of the ECG signal is that transmitting the waveform of the ECG takes up a lot of bandwidth. When several ECG waveforms are transmitted over the network, the heavy network load of the waveforms reduces throughput in the network. Additionally, high noise interference from movement of the patient results in an indistinguishable waveform created while signal processing on the sensor. Therefore, a lightweight ECG heartbeat extraction algorithm was developed to extract the heartbeat of the patient from the ECG waveform and to be resilient to noise. The ECG extraction algorithm is based on the SQRS algorithm [11], which can be found on MIT’s Physionet website [10]. The algorithm has been tailored for ECG waveforms with simulated noise superimposed on the underlying signals. The thresholding method was made more robust to noise experienced on the ECG leads. The SQRS algorithm extracts features using the derivation of the waveform to classify them. We have observed that SQRS occasionally misclassifies features due to noise. To address this issue, our implementation employs a moving threshold with lower and upper values that were independent of one another. This makes our algorithm more robust to the specific types of noise experienced on embedded ECG devices. Due to memory constraints, additional noise classification was removed from the algorithm to reduce its size.

Increasing the frequency of the ECG waveform captures more data and reveals a more precise waveform; however, if several ECG devices are in the vicinity, a lower frequency could improve the overall network capacity due to congestion; in an application such as triage, this is a valid assumption. An electronic triage system was remotely monitoring the vital signs of many patients, many ECG motes would be in close vicinity to one another.

5. EXPERIMENTATION

We used simulation to analyze the power utilization and processing delay of DYNASEC running on a Mica2 mote. We analyzed how the different levels of security and various packet sizes affect delay and power. The code was run using Avrora [16], which finds a good medium between cycle-accurate simulation and lower-granularity functional simulation. Avrora recognizes that many embedded systems go to sleep on a regular basis and triggers simulation only when an event is at the head of the queue. Avrora is cycle-accurate and models all low-level events.

DYNASEC has four levels of security: L0: plaintext; L1: MIC authentication; L2: Skipjack encryption with a 64 bit key; and L3: RC5 encryption with a 64 bit key. Each level of security has a processing delay due to encryption and decryption that increases with the security level. DYNASEC changes the amount of information that is transmitted and the type of encryption used in order to meet timing constraints in a network. DYNASEC transmits data in three sizes: 1 byte for heart rate, 50 bytes for low ECG waveform sample frequency, and 100 bytes for high ECG waveform sample frequency.

5.1 Processing Delay Measurements

The processing delays of the four security levels were measured at different packet sizes (100 bytes, 50 bytes, and 1 byte) for normalized time periods. Fig. 1 shows that modifying the packet size has a large effect on the delay. Switching from the heart rate extraction algorithm to a waveform reduces the number of packets that can be transmitted by approximately 75 fold. When the network cannot support all the data, DYNASEC selects a lower security level. In the case when one node has a large amount of high priority data, DYNASEC can redistribute the security such that all nodes meet their deadlines. By optimizing security, DYNASEC allows a healthcare provider to extract enough information to effectively monitor patients and still comply by HIPAA guidelines. Maximal security at all times would be ideal, but is unfortunately unrealistic for the current generation of wireless embedded devices employed in BANs.

5.2 Dynamic Security Allocation

Dynamic security allocation is modeled as a budgeting problem on a directed acyclic graph (DAG) representing communication links in a network. It is assumed that data is collected at sensors and routed through acyclic paths toward a centralized collection of nodes for further processing.

Let \( V = \{v_1, \ldots, v_N\} \) be the nodes in the network. Communication links in the network are organized as a DAG \( G(V, E) \), where \( V \) is the set of nodes and \( E \) is a set of edges. An edge \( (v_i, v_j) \in E \) indicates the existence of a communication link from \( v_i \) to \( v_j \). The delay of \( e \), denoted \( D(v_i, v_j) \), is the time required to transmit a packet across the link from \( v_i \) to \( v_j \). Each security algorithm \( s_j \) has an associated delay \( d_i \), the time required to encrypt the data. \( q_i \) is the probability that \( s_j \) will be broken by an adversary at a single node and \( p_i = 1 - q_i \) is the probability that \( s_j \) will not be broken. We assume that there are \( M \) different security algorithms. In the case of this paper, \( M = 4 \).

Let \( c_{ij} = 1 \) if node \( v_j \) selects security algorithm \( s_i \) and 0 otherwise. A legal solution to the problem defined above assigns exactly one security algorithm to each node in the network.

**Definition 1:** The security of a system, \( S \), is defined to be the aggregate probability that there are no security failures for node \( i \) to \( j \) for security level \( k \): \( N \) is the number of nodes in the network.

\[
S = \prod_{k=1}^{M} \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} p_{ij} \tag{1}
\]
Definition 2: A valid path $P$ is a sequence of nodes $P = <v_i, ..., v_K>$ such that there is a link from $v_i$ to $v_j$, $i \leq j \leq K$ and where the beginning node is the designated source and the ending node is the designated destination. The delay of $P$, denoted $D(P)$, is computed as follows, and includes the delay of the security algorithm $d_k$, which is performed at $v_i$. Each edge from $i$ to $j$ has $M$ security levels represented by $k$. There may exist several different paths $h$ between the source and the destination node. The delay $D$ is the summation of the processing delay $d_k$ of the security algorithm for all links on a path and all security levels. Since $c_{ijk}$ is a binary value, this results in a discrete linear program.

$$D(P) = \sum_{j=1}^{N} \sum_{k=1}^{M} c_{ijk} d_k \quad \forall h$$  

(2)

In a DAG, a source is a node with no predecessors and a sink is a node with no successors. The longest possible line of communication in a DAG is from a source to a sink. Let $P*$ be the set of all paths originating on sources and terminating on sinks. Lastly, $T$ is defined to be a global timing constraint, which must be met in order for the embedded system to transmit the necessary packets in order to respond to an anomaly.

The problem statement and formulation are as follows:

Problem Statement: Given a network modeled as a DAG $G(V,E)$, assign security algorithm $s_i$ to each node such that the overall security of the system is maximized while meeting the global timing constraint.

Problem Formulation:

Maximize: $S$  

Subject to the Following Constraints:

$$\sum_{k=1}^{M} c_{ijk} = 1 \quad \forall ij \quad c_{ij} \in \{0,1\}$$  

(4)

$$\forall P \in P* \quad D(P) \leq T$$  

(5)

$$D(v_i, v_{j+1}) \geq 0$$  

(6)

$$\sum_{k=1}^{M} c_{ijk} d_k f_{ijk} \leq l_{ij} \quad \forall ij$$  

(7)

$$f_{ij} = f_{jk}$$  

(8)

Constraint (4) ensures that exactly one security algorithm is assigned to each node. Constraint (5) ensures that each source-to-sink path in the DAG satisfies the timing constraint. Since every possible communication-path is a sub-path of some source-to-sink path, constraint (5) satisfies every possible communication path in the DAG. Constraint (6) guarantees that the delay is a positive value. Each edge from $i$ to $j$ has a link quality $l$. This link quality $l$ represents the number of packets that can make it to $j$ without retransmissions in a time period on path $h$. $O$ is the maximum number of paths that goes through link $ij$. Constraint 7 guarantees that the flows $f_{bijk}$ through link $ij$ are less than or equal to the link capacity $l_{ij}$. Constraint 8 guarantees that the flows going into node $j$ is equal to the flow exiting node $j$. A separate linear program is run for each packet size. The largest packet size is run first and if no solution exists, than a smaller packet size is used.

5.2.1 Local Decisions to Improve Propagation Delay

Currently, a central node has additional processing power and serves as a sink. The sink uses linear programming to solve the constraints using Cplex, an optimization software package. The solver will ensure that constraint (4) is satisfied and objective (3) is maximal.

However, since our system is delay sensitive, the propagation time from the central node to the nodes is undesirable. Therefore, the algorithm is not run again unless there is at least a 15% change in link quality. If there is more than a 15% change in link quality, the node temporarily makes a decision for the best path until the new solution from the central node is propagated to all the nodes in the network. When a node makes a local decision, it uses the link quality as a metric for choosing the next hop and modifies the security level according to this metric to meet timing constraint (4). In this algorithm, each node dynamically lowers its security level if it cannot achieve a desired level of throughput, and dynamically raises its security level if its current throughput level is significantly higher than the minimal threshold. Upon receiving the solution from the central node, the local node conforms to the optimal solution. Notice that this temporary solution may result in bandwidth being wasted in the overall network. Even though, modifying the packet size dynamically is essential for a functional system, it is not an optimal solution because in an optimal solution the data should be transmitted at the desired packet size from the source. Therefore, the network will temporarily be in a state of non-optimality until the optimal solution is routed to all nodes.

5.2.2 Experimentations

A central node calculates the optimal security level that each node connecting node should use. To analyze this delay, a simple network configuration was analyzed in Cplex using the formulation given above. Constraint 3 was converted to a summation for an integer linear program using the following derivation.

$$\text{Maximize } S = \log \prod_{k=1}^{N} \sum_{e_{ik}} c_{ijk}$$  

(9)

$$\log S = \log \prod_{k=1}^{N} \sum_{e_{ik}} c_{ijk}$$  

(10)

$$\text{Maximize } S = E \times \sum_{e_{ik}} c_{ijk} q_{ij}$$  

(11)

Equation (9) is similar to equation (1) except $\forall e_{ijk}$ is used instead of the summation of $i$ and $j$. Equation (10) multiples the entire equation by log and the product is consequently eliminated in equation (11). The first part of the equation reduces to $E$, the summation of all edges. From the derivation above, one can see...
that maximizing S is equivalent to minimizing the second term, the summation of the $c_{ijk}p_{ij}$.

The algorithm ran quickly in Cplex averaging 0.0015 seconds for a small number of nodes. This preliminary experimentation verifies that feasibility of the algorithm for a small number of nodes with one source and one destination node. This small number of nodes is reasonable for a small body area network, but future work includes verifying that this algorithm is scalable. The algorithm should work with multiple sources and destinations and also have a reasonable computational delay in Cplex with a larger set of nodes.

6. CONCLUSIONS

In conclusion, an authentication and encryption scheme with four levels of security was created in DYNASEC. The application layer had two distinct security modes: (1) authentication and (2) authentication and encryption. A reconfigurable security system was presented that switched between different security levels and packet sizes depending on timing constraints. DYNASEC’s architecture gives the embedded system the flexibility to adapt to the dynamic environment where it is deployed. Our reconfigurable application uses the heart rate detection algorithm to modify its packet size in response to network and system conditions. DYNASEC takes advantage of SOS’s unique architecture and use the SPI interrupt handler to send only the necessary headers over the network. Other operating systems for embedded systems, such as TinyOS, cannot dynamically change the header length without manually reprogramming the embedded system. DYNASEC’s adaptability gives systems a reasonable amount of security while meeting timing constraints.

Future work will also augment my current analysis with experimentation in a testbed. In particular, we will compare measurements done in the simulation with testbed measurements. We will also measure the delay incurred between the time that the node makes a local decision and receives the global decision from the central node. In addition, a distributed solution for dynamic security allocation on lightweight embedded systems will be investigated and the delay and efficiency of a global and distributed solution will be compared.

7. REFERENCES


