Low-Power, Secure Routing for MICA2 Mote

Breanne Duncan and David Malan

TR-06-04

2004

Computer Science Group
Harvard University
Cambridge, Massachusetts

Low-Power, Secure Routing for MICA2 Mote

Breanne Duncan
Harvard University
breanne@eecs.harvard.edu

David Malan
Harvard University
malan@eecs.harvard.edu

Abstract

Current distributed sensor network platforms lack comprehensive low-power routing techniques and efficient public key cryptography mechanisms. Reducing power for individual radio transmissions has not been explored sufficiently. Popular sensor node platforms do not include a mechanism for distributing and redistributing shared cryptographic keys among nodes. This paper discusses a technique to tailor node transmit power to the lowest practical level while maintaining reliable network links and presents the first known implementation of elliptic curve cryptography for sensor networks. Results demonstrate that dynamic radio output power scaling is effective in reducing node power consumption by orders of magnitude in certain scenarios. Analysis suggests that secret-key cryptography is already viable on the UC Berkeley MICA2 mote and public-key infrastructure may also be tractable despite the device's limited memory.

1 Introduction

Wireless sensor networks have been proposed for such applications as habitat monitoring [7], structural health monitoring [34], and vehicular tracking [49]. Of recent interest is their applicability to emergency medical care [65], a domain which demands of any technology security and longevity alike. Unfortunately, the state of the art offers weak, if any, guarantees of these needs.

The limited resources of current sensor node implementations render them ill-suited for the most straightforward implementations of these needs. The UC Berkeley MICA2 mote [13] is a low-power sensor device whose low cost can be attributed to its lack of formidable resources. It offers an 8-bit, 7.3828MHz ATmega 128L processor, 4 kilobytes (KB) of SRAM, 128 KB of program space, and 512 KB of EEPROM. The Chipcon CC1000 transceiver it employs runs at a default of 433MHz. The baud rate is 38.4K and the default, per-packet payload under TinyOS is 29 KB. The mote's size is dominated by its two AA batteries. These resources seem unfit for computationally expensive or power-intensive operations. Explicit power saving techniques are necessary to extend battery life as much as possible. Communication is much more expensive than computation on wireless sensor devices. For instance, the MICA2 radio component requires 30% more power than the CPU [28]. Low-power radio operation is necessary to carry out long-term monitoring with sensor network deployments. If the radio and CPU are constantly active, battery power will be consumed in less than a week. Device resource constraints often persuade researchers to rule out public-key cryptography as an infrastructure for authentication, integrity, privacy, and security for sensor networks [56, 64]. However, little empirical research has been published on the viability of public-key infrastructure (PKI) for sensor networks.

This work pursues low-power, secure routing for the MICA2 mote. Our implementation of the Elliptic Curve Key Agreement Scheme, Diffie-Hellman 1 (ECKAS-DH1) [46], and an analysis of another implementation of Public-Key Cryptography Standard (PKCS) #3: Diffie-Hellman Key-Agreement Standard [36], argues that public-key cryptography may be tractable on the MICA2. Instrumentation of TinyOS suggests that secret-key cryptography is already tractable on the device. We also present a power-saving approach orthogonal to existing energy-aware routing techniques that attempt to minimize radio costs by decreasing network traffic. Variable power radios such as the Chipcon CC1000 can select a minimum output power level such that messages are transmitted successfully to their destination, possibly using less power than the default setting.

This paper explores our implementation of an ECC encryption mechanism and a technique for dynamically scaling radio transmit power for the MICA2 mote. The first section deals with low-power communication techniques. Section 2.1 outlines the idea of adaptive radio power scaling as an energy-saving technique. The following subsection outlines its implementation. Section 2.3 discusses results from experiments using the algorithm in a real sensor network testbed. The second section discusses public key cryptography on the MICA2 mote. Section 3.1 analyzes TinySec, TinyOS's existing secret-key infrastructure based on SKIPJACK [10]. Section 3.2 redresses its shortcomings and examines one MICA2 implementation of Diffie-Hellman, based on the Discrete Logarithm Problem (DLP). Section 3.3 presents our implementation of Diffie-Hellman, based on the Elliptic Curve Discrete Logarithm Problem (ECDLP). Section 4 proposes directions for future research, while Section 5 explores related work. Section 6 concludes.

2 Dynamic Radio Power Scaling

Battery-powered wireless sensor devices are inherently constrained in terms of energy use. In many deployment scenarios, nodes must last months or years without external energy sources or battery replacement. For example, it is extremely impractical to periodically provide fresh batteries for thousands of environmental monitoring nodes scattered throughout a vast area. Battery life must be maximized to extend the possible uses of wireless sensor networks and reduce the burden of manual battery replacement.

The amount of power used for radio communication in wireless sensor nodes typically dominates that used in computation [29]. On the MICA2 mote, the Chipcon CC1000 radio device uses approximately 10mA while transmitting at default power (27mA at maximum), 11mA when receiving, and 8mA in idle mode [28]. The 4MHz Atmel microcontroller central to the processing unit consumes 8mA when active, but less than 15μA in sleep modes [45]. The Mica mote uses more than twice as much current under radio transmission than when using the CPU [44]. It is optimal to reduce the time the radio spends in active mode. Although the ability to use the sleep or idle modes mode depends on network and application behavior, one can assume that the device does not constantly communicate. MICA2 nodes that constantly run the radio, and thus the CPU for packet processing, last only five to six days on their power supply.

While decreasing radio duty cycle is invaluable as an energy saving technique, reducing the cost of each transmission is equally important. There exists a lower bound on the amount of communication that a given sensor network deployment requires running a certain application. Further improvement is achieved only by minimizing the current used to power an active radio. At the operating system level, strategies must be employed such that nodes can communicate successfully using the minimum output power necessary to reach one another.

Using variable power radios, nodes can adjust their power to the lowest level practical to maintain reliable connections to their neighbors. Radios with this ability are standard many sensor node devices, such as the Mica mote family. The Chipcon Ultra Low Power Transceiver on the Mica motes support a wide range of output power levels for transmission. In sufficiently dense networks, nodes may be located in close proximity to one another. This attribute may allow for a large portion of the devices to lower their radio output power and still communicate effectively with neighboring nodes. The bulk of existing power-saving techniques and protocols do not take variable radio output power strategies into account.

2.1 Adaptive Radio Power Scaling

The default output power of a given sensor node radio may be more than necessary to reliably transmit packets to a given destination. A mote can self-configure its radio power based on a comparison between target link quality and its perceived radio channel quality to the next-hop packet destination. Under the assumption that radio output power and link quality are related, decreasing power will decrease quality and increasing power will increase quality. A node uses the feedback from periodic link quality measurements to determine if more modification is necessary. Frequent link quality updates allow output power to adapt to changing environmental conditions and node movement.

Link quality can be measured a number of different ways. Appropriate measurement depends on the traffic characteristics of a given application under the behavior of a certain network stack. Signal strength can be used an an implicit indicator of link quality. Also, nodes can explicitly report to their neighbors statistics on the number of packets successfully received from each node in a given time period.

Sensor network operators or application developers may establish a target reliability representing acceptable link quality. This metric can be in terms of a percentage of packets received by a single-hop destination node. For example, it may be known that 30% general packet loss is acceptable and does not adversely effect performance of a given application.

Under the adaptive power scheme, a node changes its radio power according to observed packet loss. If link quality is higher than necessary, representing that more packets can be lost before performance is affected, the radio decreases its power. Under circumstances in which environmental obstacles and ambient noise do not heavily impact link quality, this allows link quality and radio range to decrease to the specified target while saving power. If packet loss is unacceptably high, the radio increases its power in attempt to improve link quality and allow for a larger number of successful transmissions. To prevent heavy oscillations due to frequent increase and decrease in power levels, a range of acceptable packet loss values can be implemented such that the node does not adjust its radio power within this range.

Under the adaptive scheme, node-to-node link quality optimally will be comparable to that at the default radio power, but with reduced energy consumption. Nodes close to their neighbors may lower their power significantly and still maintain as reliable a connection. Those nodes that have more difficulty reaching a neighbor with adequate signal strength may boost their output power above the default in attempt to improve link quality. Thus overall network link quality may improve, while the power consumption across the network may be equivalent to that of a fixed output power scenario.

2.2 Implementation

Ad-hoc, multi-hop routing protocols are well suited for typical sensor network. The Surge routing protocol [19] included in the TinyOS package allows sensor nodes to establish contact with a single base station, or sink node, through multiple hops. The experimental application paired with Surge transmits to the sink node light readings taken from a mote's sensor board every 8 seconds. This implementation provides a good basis for emulating a real data collection network using an ad-hoc routing protocol. It does not include any optimizations such as data aggregation or computation at intermediary nodes. Because of its simplicity and focus on transport only, it is an optimal package with which to test network communication behavior under the adaptive power scheme.

2.2.1 The Surge Routing Protocol

The Surge algorithm selects routes from a source node to the base station. The route creation phase forms a spanning tree rooted at the sink node. Each mote forwards packets only to its parent in the tree. A parent is selected from neighboring nodes with which a node is able to communicate, on the basis of link quality and hop count to the sink node. Parents may change dynamically over time based on link quality between nodes.

Each mote beacons a link quality metric for each of its neighbors every 20.5 seconds at default. The value transmitted is the number of packets the node has received, or overheard being transmitted to another node, from that neighbor. The originating node compares the number of packets it has sent to the information on packets received by a neighbor to determine packet receipt ratios. The link quality estimate is directional in that it represents the condition of the path from the child to the parent node, but not vice versa.

2.2.2 Self-Configuring Radio Output Power

The adaptive power scaling algorithm is implemented as a code module in the TinyOS platform, written in the nesC language. It is situated between the existing Surge routing protocol and the Chipcon CC1000 radio control stack. Surge uses the adaptive power module API as an interface to lower level radio commands. With a call to the module's `AdjustPower()` function, radio output power is modified according to the current Surge link quality estimate between a node and its parent. This function is called after an updated link quality estimate is received from a node's parent. If a node does not currently have a parent, it continues broadcasting light reading messages as defined by the Surge protocol, but does not modify its radio output power.

The implementation uses high and low link quality thresholds to adjust radio output power. These values have been chosen somewhat arbitrarily to designate good packet transmission rates reasonable for wireless links. Thresholds and other parameters that effect algorithm performance are displayed in Table 2.2.2. When the adaptive power algorithm is used, each node starts transmission at the default power level (1mW). An initial wait period is used to allow the network to build a routing tree to the base station before nodes adjust their radio power. Testing showed that nodes are likely to find a parent, if they are to find one under current conditions, before 45 packets are forwarded. After adjusting its power level, a node may not adjust its power level for 40 seconds to give time for link quality to react accordingly before making another change. Otherwise a node may change its power haphazardly based on inaccurate link quality information.

Parameter	Value
Initial Radio Output Power	1mW
Low Link Quality Threshold	67%
High Link Quality Threshold	78%
Initial Wait Period	45 packets
Post-Adjustment Wait Period	40 sec

Table 1: **Adaptive Algorithm Experimental Parameters.**

2.3 Results

2.3.1 Testbed

The experimental testbed for this project is "moteLab", a distributed sensor network testing environment in the Harvard Division of Engineering and Applied Sciences building. Currently the network consists of 26 MICA2 motes with attached sensor boards. Each mote is powered by a wall outlet. A node can send data packets from its UART through the building's Ethernet to be collected by a central moteLab server. These packets feed into an SQL database. Users can program nodes individually or in groups to run given applications. One can specify packet formats for the central server to collect.

The MICA2 mote uses the Chipcon SmartRF CC1000 single chip very low power transceiver, which has 23 different power levels [8]. Output power ranges from 1μW to 10 mW at the 433MHz frequency setting. However, power levels are not distributed evenly across this range. The default output power is 1mW, level 14. Available power levels are listed in [8].

All experiments were carried out using 17 moteLab MICA2 motes, the maximum number available at some points in the testing period. 12 nodes are on the second floor, 3 on the third, and 2 on the first floor. Each mote ran the Surge light sensing application using the adaptive power component. Test periods lasted an hour. Mote #4, on the second floor, acted as the Surge base station node. It was chosen for this role as an arbitrary node on the physical edge of the network, which is a common location for the data sink node.

Time scales presented here are in terms of the sequence numbers assigned to packets arriving at the moteLab server. Thus one unit as displayed on the graph does not correspond exactly to one unit of physical time, but time progression is preserved. The power axis in Figures 1 through 2 represents approximate radio output power at a given time.

2.3.2 Radio Channel Quality

The experimental phase sought to determine radio channel quality between pairs of nodes under the adaptive scheme. For data collection purposes, a node reports link quality data to the UART each time it forwards or creates a Surge light reading packet. As described in Section 2.2.1, this metric represents the percentage of packets successfully transmitted from a node to its parent. Analyzing each node's reported link quality over time shows the quality of the radio channel in response to changes in source node's output power level. Results varied greatly in terms of number of parent changes and link quality levels across time. Many nodes adapted their radio output power to a lower level without compromising connection reliability. Other nodes' link quality oscillated greatly over time.

Optimal Response

Some nodes, especially those in close proximity to their chosen parent, respond well to the adaptive power scaling algorithm. Such motes can tune their power to a level well below the default without link quality degradation. An example is shown in Figure 1. Packets are transmitted to the parent node at low power, but are successfully received due to favorable environmental conditions and parent proximity. For instance, mote #7 uses 8 times and 99 times less power under the adaptive strategy than under power levels 14 (default) and 4, respectively. However, packet reception rates are excellent in each case. The adaptive technique yields a 91% packet reception rate, while the two other levels yield 93%, a negligible difference.

Link Quality Oscillations

As certain nodes lower their output power, link quality decreases sharply as a result. In many cases the packet reception beaconed by a parent node notifies the child of a link quality drop of greater than 50% as compared with the last broadcast sent. As shown in Figure 2, the given node cannot respond to the degradation until after it has been reported. Thus it is unable to prevent a drop of such great magnitude.

Large oscillations in radio channel quality occur as a result of this process. The child node reacts to the link quality degra-

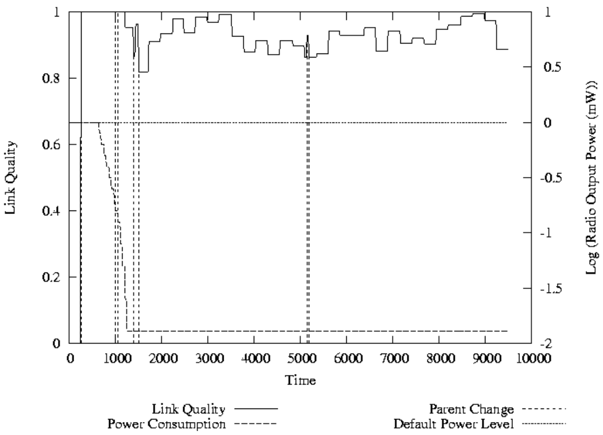

Figure 1: **Optimal Dynamic Power Scaling Behavior.** *Some nodes are able to decrease their power use immensely with little or no link quality degradation.*

dation by increasing power sufficiently. Link quality increases immediately as a result and the node once again calculates that it may decrease its out power level. Upon decreasing output power, link quality drops substantially and the process repeats.

In the oscillatory case, radio output power is usually less under the adaptive algorithm than at the default power level. However, if a lower fixed power level is used, approximately the same amount of power may be consumed while preventing the oscillations that decrease overall link quality across time. For example, in an adaptive run, node 13 has 67% link quality with its parent. When using power level 5 constantly, however, it increases its connection quality by 21% and decreases power use by 12%. Deep oscillations are not present.

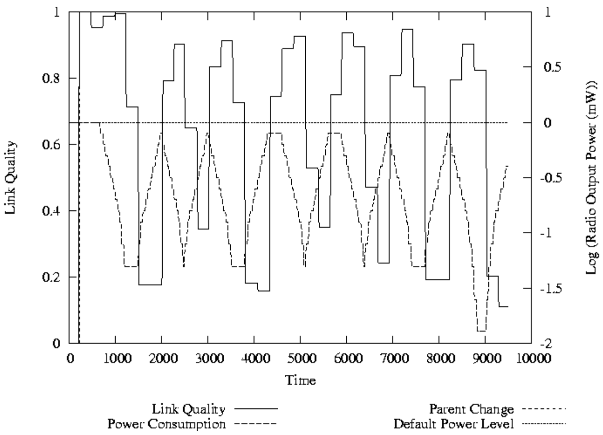

Figure 2: **Link Quality Oscillations.** *Cycles of link quality degradation and improvement result from the adaptive output power algorithm for some nodes.*

Several techniques may help dampen large link quality oscillations. Nodes can send Surge link quality beacons more frequently. Intermediary link quality estimates may show a more graceful degradation to which nodes can react earlier, preventing immense link quality drops. However, transmitting this information more often requires a higher radio duty cycle and thus more energy. It was conjectured that perceived drops in link quality were often caused by some link quality beacons simply not reaching the node. Thus the node would receive a high link quality estimate from its parent and later receive a very low value, missing those in between that would presumably have intermediary values. To test this theory, nodes beaconed all link quality estimates at the default power level, but continued to send light reading packets at the radio power specified by the adaptive algorithm. In theory, link quality values would reach the child nodes, who could increase their power after a smaller link quality reduction. However, testing proved this method ineffective, disproving the conjecture.

Theoretically, a node may prevent oscillations by keeping its radio output power above the level that last caused a sharp drop in link quality. A node can calculate the difference between old and new link quality beacon values. If the quality has decreased significantly, a node sets its minimum power level at one above the current level. In theory this should prevent future oscillations, assuming the node is relatively stationary and environmental conditions do not change drastically. If the node changes parents under the Surge protocol, the minimum is reset to the radio's physical minimum output power and the process repeats. A node can adapt its minimum power level to each parent. In a mobile network, the minimum may be reset at intervals to ensure it is properly configured to current network conditions. In a few cases, this procedure worked successfully. However, all tests did not produce favorable results. This may be due to variances in link quality that occur regardless of a node's output power level (see Section 2.3.4).

2.3.3 Base Station Packet Reception

Because the Surge protocol facilitates single base station data collection, the percentage of packets that arrive at the sink node gives an indication of network link quality and connectivity. Table 2.3.3 gives numerical data for comparing adaptive algorithm performance to fixed power runs with similar power use. Tests were distributed across different times of day during varying human activity levels in the building. Energy statistics given here represent the summation of radio output power for each packet transmission. This metric is summed across an intermediary time interval, $2000 \leq t \leq 4000$. Omitting an initial time period helps to assess the adaptive algorithm after setup has occurred. Nodes use the default power during the startup phase, which increases power use initially, but helps establish a routing table before nodes adjust their radio power level. Log base 10 is used for plotting power values.

Table 2.3.3 shows that the adaptive algorithm is competitive with a fixed power level strategy. However, it frequently performs slightly worse as compared to fixed power levels with nearly equivalent power consumption. Under the current adaptive algorithm implementation, certain internode connections perform poorly, as discussed in Section 2.3.2. This may hinder sink node packet collection significantly. Whole network performance, in terms of packets received, drops. Thus measuring adaptive algorithm performance through base station packet reception for all nodes may be misleading.

Power Level	% Packets Received	Power Use (mW)
3	0%	210
4	55.0%	376
adaptive$_1$	63.8%	466
adaptive$_2$	73.2%	504
adaptive$_3$	74.7%	570
5	78.2%	593
adaptive$_4$	73.0%	604
adaptive$_5$	63.4%	696
adaptive$_6$	52.5%	719
6	81.6%	744
adaptive$_7$	74.2%	773
7	77.3%	941
8	66.5%	1177
adaptive$_8$	75.7%	1220
adaptive$_9$	74.7%	1245
9	66.5%	1488
14	78.1%	4707
19	79.8%	18825

Table 2: **Base Station Packet Reception and Power Use for Selected Output Power Levels and Adaptive Algorithm Tests.**

Route-Level Analysis

Viewing connection quality on the level of a mote's path to the base station provides better insight to individual node performance. This mid-level approach keeps node performance in the context of the routing tree, where ancestor link quality determines base station packet reception. The behavior of both child and parent nodes changing power is taken into account, while leaving out links completely unrelated to a node's path to the base station. However, links on the node-to-sink route that perform poorly under the adaptive strategy still affect sink node packet collection.

Figures 3-5 compare route-level performance and power use under the adaptive power scaling algorithm to a lower, fixed power level and the default output power. A test run of each power level performing relatively well was chosen. All nodes use less power than the default when dynamically scaling their radio power. However, the base station receives a smaller percentage of packets for some nodes. Motes 1, 2, 13, and 17 fall in this category. While path quality is degraded slightly, power savings are obvious for these nodes. The success of mote 14 in the adaptive graph can be attributed to favorable environmental conditions allowing the mote to find a parent. When compared to fixed power level 6, some nodes using the adaptive scheme expend more power per packet received and some consume less power. Nodes 2, 7, and 11 have higher base station packet reception per unit of power use in the dynamic power scaling scheme. However, nodes 1, 5, 12, and 16 benefit from using a fixed low power under this metric. Under the adaptive technique, more packets are transmitted successfully from these nodes to the base station. However, this increased throughput comes at the cost of significantly more power. If a target link quality is to be met, however, such a strategy prevails above using a fixed, network-wide, lower power level.

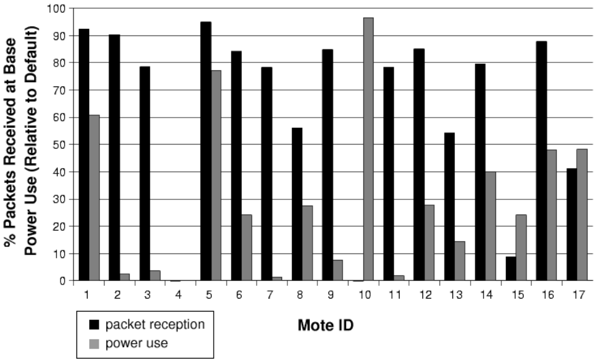

Figure 3: **Packet Reception Using Dynamic Power Scaling.**

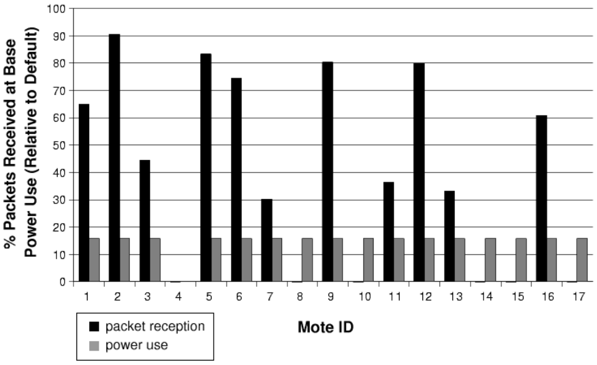

Figure 4: **Packet Reception at Power Level 6.**

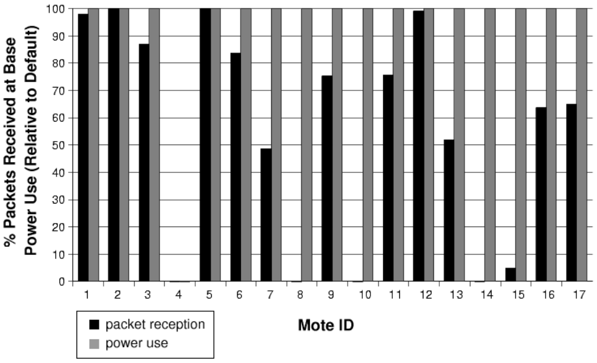

Figure 5: **Packet Reception at Default Power.**

2.3.4 *Variance*

Even at a fixed power level, quality of a single radio channel can vary greatly in a stationary, undisturbed network. The standard deviation of network connectivity was extremely high across several runs of the Surge application at a constant power level. During four different hour-long tests at power level 5, 26%, 41%, 48%, 67%, and 78% of packets were delivered to the base station on each respective run. The 26% and 67% datasets were gathered at nearly the same time of day. The 41% and 78% figures were gathered during nighttime testing.

The time at which an experiment runs affects network quality. Daytime experiments are affected by human activity in an indoor setting and sunlight. Device interference may also affect measurements. Under the dynamic power scaling strategy, nodes conjunctively delivered an average of 18% more packets to the base station at night. Like experiments at fixed power level 6 showed that an average of 18% more packets were delivered at night as well. Three tests were taken during each time period for each power use strategy. For this study, every effort was made to run comparative tests at a similar time of day.

We ran several tests to compare the performance of the adaptive algorithm with and without TinySec encryption. Results were inconclusive, most likely due to the natural variance in radio channel quality.

3 Public Key Cryptography on the MICA2

3.1 SKIPJACK

TinyOS currently offers the MICA2 access control, authentication, integrity, and confidentiality through TinySec, a link-layer security mechanism based on SKIPJACK in CBC mode. An 80-bit symmetric cipher, SKIPJACK is the formerly classified algorithm behind the Clipper chip, approved by the National Institute for Standards and Technology (NIST) in 1994 for the Escrowed Encryption Standard [50]. Through use of a shared, group key does TinySec provide for access control; with message authentication codes does it provide for messages' authentication and integrity; and with encryption does it provide for confidentiality.

Unfortunately, TinySec's reliance on shared keys render the mechanism particularly vulnerable to attack. After all, the MICA2 is intended for deployment in sensor networks. For reasons of cost and logistics, long-term physical security of the devices is unlikely. Compromise of the network, therefore, reduces to compromise of any one node.

But the mechanism is not without value. After all, it does offer an 80-bit key space, known attacks on which involve 2^{79} operations on average (assuming SKIPJACK isn't reduced from 32 rounds [4]). And, as packets with TinySec include a 4-byte message authentication code (MAC), the probability of blind forgery is 2^{-32}. This security comes at a cost of just five bytes: whereas transmission of some 29-byte plaintext and its cyclic redundancy check (CRC) requires a packet of 36 bytes, transmission of that plaintext's ciphertext and MAC under TinySec requires a packet of only 41 bytes, as the mechanism borrows TinyOS's fields for Group ID (TinyOS's weak, default mechanism for access control) and CRC for its MAC.

Meanwhile, the impact of TinySec on the MICA2's performance appears reasonable. On first glance, it would appear that TinySec adds under 2 ms to a packet's transmission time, as per Figure 6, and under 5 ms to a packet's round-trip time (for packets echoed back to their source by some neighbor), as per Figure 7. However, the apparent overhead of TinySec, as suggested by transmission times, is nearly the data's root mean squared. Though the round-trip times exhibit less variance, additional benchmarks seem in order for TinySec's accurate analysis. Figure 8, then, offers results of yet less variance from finer instrumentation of TinySec: encryption of a 29-byte, random payload requires 2,190 μs on average, and computation of that payload's MAC requires 3,049 μs on average; overall, TinySec adds $5,239 \pm 18$ μs to a packet's computational requirements. It appears, then, that some of those cycles can be subsumed by delays in scheduling and medium access, at least for applications not already operating at full duty. Figure 9, the results of an analysis of the MICA2's maximal throughput, without and with TinySec enabled, puts the mechanism's computational overhead for such applications into perspective: on average, TinySec may lower maximal throughput of acknowledged packets by only 0.29 packets per second.

	without TinySec	with TinySec	Difference
Median	72,904 μs	74,367 μs	1,463 μs
Mean	74,844 μs	76,088 μs	1,244 μs
Standard Deviation	24,248 μs	24,645 μs	n/a
Standard Error	767 μs	779 μs	1,093 μs

Figure 6: Transmission time for the MICA2, computed over 1000 trials, where transmission time is defined here as the time elapsed between `SendMsg.send(·,·,·)` and `SendMsg.sendDone()` for the transmission of a 29-byte, random payload.

Of course, TinySec's encryption and authentication does come at an additional cost. Per Figure 13, TinySec adds 3,352 B collectively to an application's data and text segments, 454 B to an application's BSS segment, and 92 B to an application's maximal stack size during execution. For applications that don't require the entirety of the MICA2's 128 KB of program memory and 4 KB of SRAM, then, TinySec seems a viable addition.

Unfortunately, the problem of shared keys remains. Pairwise keys among n nodes would certainly provide some defense against compromises of individual nodes. But n^2 80-bit keys would more than exhaust a node's SRAM for n as small as 20. A more sparing use of shared keys is in order, but secure, dynamic establishment of those keys, particularly for networks in which the positions of sensors may be transient, requires a chain or infrastructure of trust. In fact, the very design of TinySec requires as much for rekeying as well. Though TinySec's 4-byte initialization vector (IV) allows for secure transmission of some message 2^{32} times, that bound may be insufficient for embedded networks whose lifespans require larger IVs. Needless to say, TinySec's reliance on a single, shared key prohibits the mechanism from securely rekeying itself.

Fortunately, these problems of shared keys' distribution and redistribution are redressed by public-key infrastructure. The sections that follow thus explore that infrastructure's design and implementation on the MICA2.

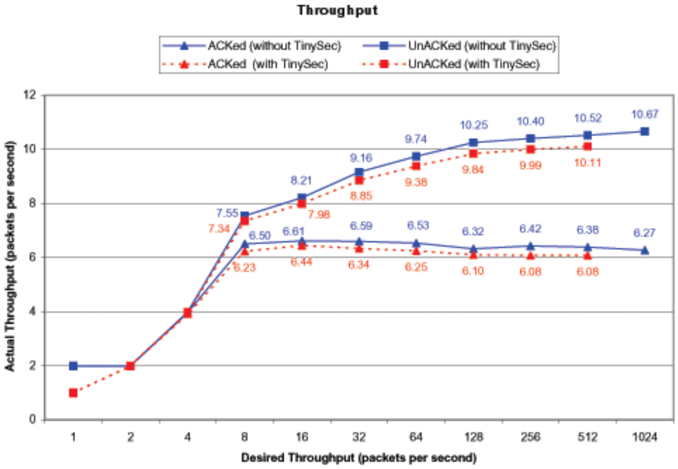

Figure 9: Actual throughput versus desired throughput for acknowledged (ACKed) and unacknowledged (unACKed) transmissions between a sender and a receiver, averaged over 1000 trials per level of desired throughput, where desired throughput is the rate at which calls to `SendMsg.send(·,·,·)` were scheduled by `Timer.start(·,·)`. ACKed actual throughput is the rate at which 29-byte, random payloads from a sender were received and subsequently acknowledged by and an otherwise passive recipient. UnACKed actual throughput is the rate at which the sender actually sent such packets, acknowledged or not (*i.e.*, the rate at which calls to `SendMsg.send(·,·,·)` were actually processed). For clarity, where ACKed and unACKed throughput begins to diverge are points labelled with values for actual throughput.

	without TinySec	with TinySec	Difference
Median	145,059 μs	149,290 μs	4,231 μs
Mean	147,044 μs	152,015 μs	4,971 μs
Standard Deviation	30,736 μs	31,466 μs	n/a
Standard Error	972 μs	995 μs	1,391 μs

Figure 7: Round-trip time for the MICA2, computed over 1000 trials, where round-trip time is defined here as the time elapsed between `SendMsg.send(·,·,·)` and `ReceiveMsg.receive(·)` for the transmission of a 29-byte, random payload and subsequent receipt of the same.

	`encrypt()`	`computeMAC()`	Sum
Median	2,189 μs	3,038 μs	5,233 μs
Mean	2,190 μs	3,049 μs	5,239 μs
Standard Deviation	3 μs	281 μs	281 μs
Standard Error	0 μs	9 μs	9 μs

Figure 8: Computational overhead of TinySec, computed over 1000 trials, where `encrypt()` denotes the time required to encrypt a 29-byte, random payload, and `computeMAC()` denotes the time required to compute that payload's MAC.

Memory Overhead of TinySec

	without TinySec	with TinySec	Difference
ROM	9,224 B	16,576 B	7,352 B
RAM	384 B	838 B	454 B
Stack	105 B	197 B	92 B

Figure 10: Results from instrumentation of CntToRfm, an application which simply broadcasts a counter's values over the MICA2's radio. Here and hereafter, ROM is defined as application's data and text segments; RAM is defined here as application's BSS segment; stack is defined here as the maximum of the application's stack size during execution.

3.2 DLP

With the utility of SKIPJACK-based TinySec thus motivated and the mechanism's costs exposed, this work turns to DLP, on which Diffie-Hellman [15] is based, as the foundation for one possible answer to the MICA2's problems of shared keys' distribution and redistribution. DLP typically involves recovery of a $x \in \mathbb{Z}_p$, given p, g, and $g^x \pmod{p}$, where p is a prime integer and g is a generator of $\mathbb{Z}_p$. By leveraging the presumed difficultly of DLP, Diffie-Hellman allows two parties to agree, without prior arrangement, upon a shared secret, even in the midst of eavesdroppers, with perfect forward secrecy. Authenticated exchanges are possible with the station-to-station protocol (STS) [16], a variant of Diffie-Hellman.

With a form of Diffie-Hellman, then, could two nodes thus establish a shared secret for use as TinySec's key. At issue, though, is the cost of such establishment on the MICA2.

Inasmuch as the goal at hand is distribution of 80 bits of security, any mechanism of exchange should provide at least as much security. According to NIST, then, the MICA2's implementation of Diffie-Hellman should employ a modulus, p, of at least 1024 bits and an exponent (*i.e.*, private key), x, of at least 160 bits [52], per Figure 11.

Unfortunately, on an 8-bit architecture, computations with 160-bit and 1024-bit values are not inexpensive. However, modular exponentiation does not appear to be intractable on the MICA2. Figure 12 offers the results of instrumentation of one implementation of Diffie-Hellman for the MICA2 [63]: computation of $2^x \pmod{p}$, where x is a 160-bit integer and p is a well-known, 768-bit prime, requires 31.0 s on average; computation of the same, where p is a well-known, 1,024-bit prime, requires 54.9 s. Assuming a node need only be rekeyed, on average, every 2^{32} packets (at which time its initialization vectors are exhausted), this computation and that for $y^x \pmod{p}$, where

Bits of Security	Modulus	Exponent
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

Figure 11: Strength (*i.e.*, bits of security) of Diffie-Hellman for various moduli and exponents. [52]

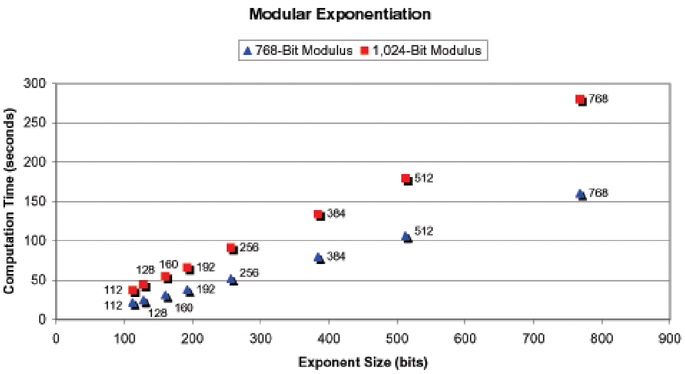

Figure 12: Time required to compute $2^x \pmod{p}$, where p is a well-known prime, on the MICA2.

y is another node's public key, may be acceptable costs for an application's longevity.

Of course, these computations require that the MICA2 operate at full duty cycle, the power requirements of which may be unacceptable. After all, although the theoretical lifetime of the MICA2, powered by two AA batteries, is as many as twenty years at minimal duty cycle (assuming no self-discharge), that lifetime decreases to just a few days at maximal duty cycle. Figure 14 reveals the power consumption of modular exponentiation on the MICA2: computation of $2^x \pmod{p}$ appears to require 1.185 J. Roughly speaking, a mote could devote its lifetime to 51,945 such computations.[1]

Unfortunately, these computations require not only time but also memory. Mere storage of a public key requires as many bits as is the modulus in use. Accordingly, n 1,024-bit keys would more than exhaust a node's SRAM for n as small as 32. Although a node is unlikely to have—or, at least, need—so many neighbors or certificate authorities for whom it needs public keys, Diffie-Hellman's relatively large key sizes are unfortunate in the MICA2's resource-constrained environment.

But, through elliptic curve cryptography (ECC), 80 bits of security may be available to the MICA2 at a lower price than 1,024 bits: 163 bits. Indeed, elliptic curves are thought to offer computationally equivalent security with remarkably smaller key sizes insofar as subexponential algorithms exist for DLP [1, 20, 57, 37], but no such algorithm is known or thought to exist for ECDLP over certain fields [18, 11].

	768-Bit Modulus	1,024-Bit Modulus
ROM	2186 B	2234 B
RAM	467 B	595 B
Stack	136 B	136 B

Figure 13: Memory usage of an implementation of modular exponentiation on the MICA2 which computes $2^x \pmod{p}$, where x is a 512-bit integer and p is a well-known prime.

	1,024-Bit Modulus, 160-Bit Exponent
Average Current	7.3 mA
Total Time	54.1144 s
Total CPU Utilization	3.9897×10^8 cycles
Total Energy	1.185 J

Figure 14: Power consumption of one execution of an implementation of modular exponentiation on the MICA2 which computes $2^x \pmod{p}$, where x is a 512-bit integer and p is a well-known prime.

3.3 ECDLP

Elliptic curves offer an alternative foundation for the exchange of shared secrets among eavesdroppers with perfect forward secrecy. ECDLP, on which ECC [47, 32] is based, typically involves recovery over some Galois (*i.e.*, finite) field, $\mathbb{F}$, of $k \in \mathbb{F}$, given (at least) $k \cdot G$, G, and E, where G is a point on an elliptic curve, E, a smooth curve of the long Weierstrass form

$$y^2 + a_1xy + a_3y \equiv x^3 + a_2x^2 + a_4x + a_6, \tag{1}$$

where $a_i \in \mathbb{F}$. Of particular interest to cryptographers are $\mathbb{F}_p$ and $\mathbb{F}_{2^p}$, where p is prime, as neither appears vulnerable to subexponential attack [18]. Though once popular, extension fields of composite degree over $\mathbb{F}_2$ are vulnerable by reduction with Weil descent [17] of ECDLP to DLP over hyperelliptic curves [18]. But $\mathbb{F}_{2^p}$, a binary extension field, remains popular among implementations of ECC, especially those in hardware, inasmuch as it allows for particularly space- and time-efficient algorithms. In light of its applications in coding, the field has also received more attention in the literature than those of other characteristics [53].

It is with this history in mind that I proceeded with my first, and, later, second, implementation of ECC over $\mathbb{F}_{2^p}$ toward an end of smaller public keys for the MICA2. Background for these implementations' designs now precedes their results.

3.3.1 *Elliptic Curves over* $\mathbb{F}_{2^p}$

It turns out that, over $\mathbb{F}_{2^p}$, Equation 1 simplifies to

$$y^2 + xy \equiv x^3 + ax^2 + b, \tag{2}$$

where $a, b \in \mathbb{F}_{2^p}$, upon substitution of $a_1^2x + \frac{a_3}{a_1}$ for x and $a_1^3y + \frac{a_1^2a_4 + a_3^2}{a_1^3}$ for y, if we consider only nonsupersingular curves, for which $a_1 \neq 0$. It is the set of solutions to Equation 2 and, more generally, Equation 1 (*i.e.*, the points on E), that actually provides the foundation for smaller public keys on the MICA2. All that remains is specification of some algebraic structure over that set. An Abelian group suffices but requires provision of some binary operator offering closure, associativity, identity, inversion, and commutativity. As suggested by ECDLP's definition, that operator is to be addition.

[1] According to [9], Energizer No. E91, an AA battery, offers an average capacity of 2,850 mAh. It follows that 2 × 2,850 mAh × 3600 s/h ÷ (7.3 mA × 54.1144 s) ≈ 51,945 modular exponentiations may be possible with two AA batteries on the MICA2.

The addition of two points on a curve over $\mathbb{F}_{2^p}$ is defined as $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$, where

$$(x_3, y_3) = (\lambda^2 + \lambda + x_1 + x_2 + a,\ \lambda(x_1 + x_3) + x_3 + y_1),$$

where $\lambda = (y_1 + y_2)(x_1 + x_2)^{-1}$. However, so that the group is Abelian, it is necessary to define a "point at infinity," $\mathcal{O}$, whereby

$$\begin{aligned} \mathcal{O} + \mathcal{O} &= \mathcal{O}, \\ (x, y) + \mathcal{O} &= (x, y), \text{ and} \\ (x, y) + (x, -y) &= \mathcal{O}. \end{aligned}$$

Doubling of some point, meanwhile, is defined as $(x_1, y_1) + (x_1, y_1) = (x_3, y_3)$, where

$$(x_3, y_3) = (\lambda^2 + \lambda + a,\ x_1^2 + (\lambda + 1)x_3),$$

where $\lambda = x_1 + y_1 x_1^{-1}$, provided $x_1 \neq 0$.

With these primitives is point multiplication also possible [21]. With an algebraic structure on the points of elliptic curves over $\mathbb{F}_{2^p}$ thus defined, implementation of a cryptosystem is now possible.

3.3.2 ECC over $\mathbb{F}_{2^p}$

Implementation of ECC over $\mathbb{F}_{2^p}$ first requires a choice of basis for points' representation, insofar as each $a \in \mathbb{F}_{2^p}$ can be written as $a = \sum_{i=0}^{m-1} a_i \alpha_i$, where $a_i \in \{0, 1\}$. Thus defined, a can be represented as a binary vector, $\{a_0, a_1, \dots, a_{p-1}\}$, where $\{\alpha_0, \alpha_1, \dots, \alpha_{p-1}\}$ is its basis over $\mathbb{F}_2$. Most common for bases over $\mathbb{F}_2$ are polynomial bases and normal bases, whereby the former tends to be more efficient in software [3], though dual, triangular, and other bases exist. Admittedly, polynomial bases are also simpler conceptually and, thus, daresay, an apt choice for a first implementation of ECC on the MICA2.

When represented with a polynomial basis, each $a \in \mathbb{F}_{2^p}$ corresponds to a binary polynomial of degree less than p, whereby $a = a_{p-1}x^{p-1} + a_{p-2}x^{p-2} + \cdots + a_0 x^0$, where, again, $a_i \in \{0, 1\}$. Accordingly, each $a \in \mathbb{F}_{2^p}$ will be represented in the MICA2's SRAM as a bit string, $a_{p-1}a_{p-2}\cdots a_0$. All operations on these elements are performed modulo an irreducible reduction polynomial, f, of degree p over $\mathbb{F}_2$, such that $f(x) = x^p + \sum_{i=0}^{p-1} f_i x_i$, where $f_i \in \{0, 1\}$ for $i \in \{0, 1, \dots, p-1\}$. Typically, if an irreducible trinomial, $x^p + x^k + 1$, exists over $\mathbb{F}_{2^p}$, then $f(x)$ is chosen to be that with smallest k; if no such trinomial exists, then $f(x)$ is chosen to b a pentanomial, $x^p + x^{k_3} + x^{k_2} + x^{k_1} + 1$, such that k_1 is minimal, k_2 is minimal given k_1, and k_3 is minimal given k_1 and k_2 [40].

In polynomial basis, addition of two elements, a and b is defined as $a + b = c$, where $c_i \equiv a_i + b_i \pmod 2$ (*i.e.*, a sequence of XORs). Multiplication of a and b, meanwhile, is defined as $a \cdot b = c$, where $c(x) \equiv (\sum_{i=0}^{p-1} a_i x^i)(\sum_{i=0}^{p-1} b_i x^i) \pmod{f(x)}$. With a polynomial basis selected, all that remains is execution of this design on the MICA2.

3.3.3 EccM 1.0

Version 1.0 of EccM, my first attempt at an implementation of ECC on the MICA2 in the form of a TinyOS module, is a partial success. Designed for execution on a single mote, this version of EccM first selects a random curve in the form of Equation 2, such that $a = 0$ and $b \in \mathbb{F}_{2^p}$. It next selects from that curve a random point, $G \in \mathbb{F}_{2^p} \times \mathbb{F}_{2^p}$. It further selects at random some $k \in \mathbb{F}_{2^p}$, the node's private key. Finally, it computes $k \cdot G$, the node's public key. The running time of these operations is then transmitted to the node's UART.

Based upon code by Michael Rosing [58], EccM 1.0 employs a number of optimizations. Addition of points is implemented in accordance with [59]; multiplication of points follows [33]; conversion of integers to non-adjacent form is accomplished as in [62]. Generation of pseudorandom numbers, meanwhile, is achieved with [42].

On first glance, the results, offered in Figure 15, are encouraging, with 33-bit keys requiring a running time of just 1.776 s. Unfortunately, for larger keys (*e.g.*, 63-bit), the module fails to produce results, instead causing the mote to reset cyclically. Though this behavior appears to be undocumented [12], it seems the result of stack overflow. Although none of EccM's functions are recursive, several utilize a good deal of memory for multiword arithmetic. In fact, Figure 16 offers the results of an analysis of EccM 1.0's usage of SRAM. Unfortunately, for keys of 63 bits or more, EccM 1.0 exhausts the MICA2's SRAM.

Insofar as optimizations of EccM 1.0 fail to render generation of 63-bit keys possible, an overhaul of this first implementation proves necessary. With EccM 2.0 do I achieve my goal of 163-bit security.

3.3.4 EccM 2.0

Based upon the design of Dragongate Technologies Limited's Java-based jBorZoi 0.9 [39], EccM 2.0 similarly implements ECC but with greater success than EccM 1.0. Again a TinyOS module, EccM 2.0 selects for a node at random, using a polynomial basis over $\mathbb{F}_{2^p}$, a private key, thereafter computing with a curve and base point recommended by NIST the node's public key, the running time of which is then transmitted to the node's UART. In this version, multiplication of points is achieved with Algorithm IV.1 in [5]. Multiplication of elements in $\mathbb{F}_{2^p}$, meanwhile, is implemented as Algorithm 2 in [25], while inversion is implemented as Algorithm 8 in the same.

Beyond rendering 163-bit keys feasible, EccM 2.0 also redresses another shortcoming in 2.0. Inasmuch as 1.0 selects curves at random, it risks (albeit with exponentially small probability) selection of supersingular curves which are vulnerable to sub-exponential attack via MOV reduction [43] with index-calculus methods [60]. EccM 2.0 thus obeys NIST's recommendation for ECC over $\mathbb{F}_{2^p}$ [51].

Unfortunately, although EccM 2.0 employs less memory than EccM 1.0, per Figure 17, its running time is disappointing. The time required to generate a private and public key pair with this module, averaged over 100 trials, is 466.9 s, with a standard deviation of 16.1 s. Such performance is likely unacceptable for most applications. Moreover, the module's consumption of power is significant, as per Figure 18: generation of the node's private key requires approximately 5.7 mJ, whereas computation of the node's public key requires approximately 12.6402 J. In contrast with its calculation of $2^x \pmod p$, it appears the MICA2 could devote its lifetime to, approximately, just 4,868 of these computations.[2]

[2] This estimate follows from 2 × 2,850 mAh × 3600 s/h ÷ (8.8 mA ×

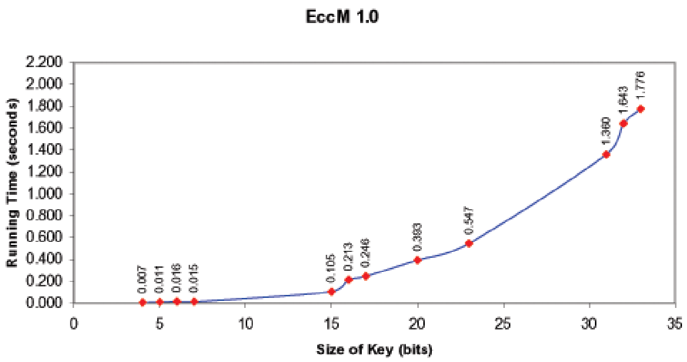

Figure 15: Running time for EccM 1.0. Points are labelled with running times. For larger keys (*e.g.*, 63-bit), the module failed to produce results.

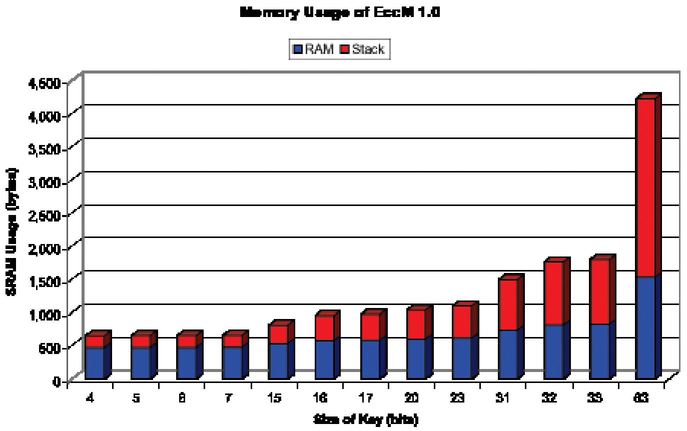

Figure 16: Memory usage of EccM 1.0. Keys of 63 bits or more exhaust the MICA2's 4,096 KB of SRAM.

With ECC thus implemented on the MICA2 in EccM 2.0, future work nonetheless remains, the most obvious of which is reduction of this implementation's running time.

4 Future Work

Future research will expand the cryptographic options and improve upon dynamic power scaling for the MICA2 sensor node. This work provides the foundation for what will hopefully become TinyCrypt, a cryptographic library for the MICA2 with support for symmetric and asymmetric keys. Many strategies may improve the effectiveness of the adaptive power scaling algorithm presented here. A variety of different adaptive algorithms may be useful. Selecting which to employ depends on the characteristics of the radio channel link between two nodes. Also, the interaction between the underlying Surge routing protocol and the dynamic output power module needs to be explored. The algorithm may benefit from certain changes of the Surge protocol.

Nodes react very distinctly under the algorithm depending on location and environmental characteristics, among other variables. Nodes with different link quality characteristics may need to use different dynamic output power configuration strategies.

	163-Bit Key
ROM	43,286 B
RAM	820 B
Stack	81 B

Figure 17: Memory usage of EccM 2.0.

	Private-Key	Public-Key
Average Current	8.8 mA	8.5 mA
Time	216.597 ms	495.70 s
CPU Utilization	1.597×10^6 cycles	3.65×10^9 cycles
Energy	5.7 mJ	12.6402 J

Figure 18: Power consumption EccM 2.0's generation of one private key and one public key.

Optimally nodes could characterize the observed radio channel behavior as being a certain type of predefined pattern. Each case would use a unique adaptive algorithm. For instance, a node far from its neighbors should not adjust its power in the same manner as a node with oscillating radio channel quality. A node unable to find a parent may not benefit from simply constantly increasing its power level until one is found. Perhaps it will find a parent after some time by periodically sending out special beacons at a high power level to probe for available neighbors without constantly expending power. A node displaying an oscillatory link quality pattern may benefit from certain techniques in addition to setting a minimum power level after quality drops significantly. Sending link quality beacons more frequently may decrease oscillations. This strategy, and the tradeoff between the resultant link quality improvement and the extra power use, has not been investigated.

Because the adaptive power scaling algorithm was added to an existing routing protocol, performance is dependant on the characteristics of the Surge protocol. Modifying the basic protocol to work efficiently in conjunction with the algorithm may help improve network communications. In the Surge algorithm, the number of parent link quality estimates received in a given time period may have an effect on the link quality metric itself. If the child receives less than 40% of the beacons, the link quality value is halved. Default adaptive algorithm reaction to this drop may not be appropriate. Increasing transmit power on the side of the child node does not improve parent beacon reception. Perhaps a new parent search should be initiated or power should be held constant as to not waste power on a link marred by adverse environmental conditions. Moreover, link quality drops resulting from oscillations or other side effects of the adaptive algorithm may cause parent changes. More investigation may show that the parent selection strategy needs to change with the addition of the adaptive algorithm to Surge.

EccM 3.0 will incorporate a number of optimizations into the source of EccM 2.0. Perhaps of greatest benefit will be 3.0's version's re-write in AVR assembly of the 2.0's multi-precision arithmetic routines, to which many of the module's other functions ultimately reduce. Like C, NesC hides hardware's carry bits and overflow flags and impedes particularly efficient execution of multi-word additions, multiplications, and shifts, all of which are crucial to ECC's performance.

Although EccM 1.0 and 2.0 already incorporate a number of published algorithms, I will return to the literature for alterna-

216.597 ms + 8.5 mA × 495.70 s) ≈ 4,868.

tives for 3.0's design. And, perhaps more drastically, I will consider a normal basis for 3.0's arithmetic, the advantage of which is its implementation using only ANDs, XORs, and cyclic shifts, beneficiaries of which are multiplication and squaring. Of value to 3.0 as well might be a hybrid of polynomial and normal bases, as such is thought to leverage advantages of each simultaneously [58].

Of course, I could reconsider 3.0's choice of fields, opting instead to implement ECC over $\mathbb{F}_p$. Unfortunately, inversion, as required by point arithmetic in this field, is not inexpensive. But the operation can be avoided through use of projective (as opposed to affine) coordinates [22]. Although relatively efficient algorithms exist for modular reduction (*e.g.*, those of Montogomery [48] or Barrett [2]), selection of a generalized Mersene number for p would also allow modular reduction to be executed as a more efficient sequence of three additions (mod p) [61].

Contingent on its optimization, EccM 3.0 might incorporate support for larger keys, particularly those sizes recommended by NIST [51], as well as pseudorandom generation of curves and base points in lieu of its reliance on NIST's samples. If a success, EccM 3.0 will provide the foundation for TinyCrypt 1.0's distribution and redistribution of keys.

5 Related Work

5.1 Low-Power Routing

Many strategies for reducing whole network or individual node power consumption in wireless sensor networks focus on reducing internode communication. Efficient data dissemination, collection, and node coordination is central to the problem. Route discovery and maintenance algorithms also play an important role in reducing communications overhead. Both the resultant routing paths and the messages sent to establish these paths inherently affect network power consumption.

Many wireless routing protocols are designed with basic energy efficiency in mind. Ad-hoc on-demand distance vector routing (AODV) is a multi-hop protocol oriented towards mobile networks [55]. Because it constructs routes on demand, it avoids the communication cost of building paths nodes may never use. Dynamic source routing (DSR) is another popular multi-hop, on-demand routing protocol [30]. DSR takes specific steps to avoid unnecessary traffic when updating routes. While AODV and DSR aim for efficient communication, they are designed for systems using 802.11 wireless components in which battery life is not largely dependent on communication. Although both have been ported to the TinyOS platform, their lack of major power-saving mechanisms undermine their viability in long-term sensor network deployments.

Cluster-based routing protocols such as LEACH [27] attempt to balance communication load by rotating data collection and aggregation duties among network nodes. An intermediary sink node, or cluster head, must keep its radio active a larger portion of the time to receive packets from neighbors. Sharing this duty prolongs sensor network lifetime by preventing cluster head burnout. This approach can be used in conjunction with dynamic radio output power scaling for increased energy savings.

Kubisch et al. define a strategy to select a minimum transmission power for each node in a given distributed sensor network [35]. Network operators define a target number of reachable neighbors for all nodes. A node adjusts its transmit quality according to its current neighbor count. If a node has more neighbors than necessary, it scales down its transmit power until reaching the target range. If too few neighbors exist, the node increases its power. This technique is not optimal for data collection networks. It does not take into account node hop distance from a data sink node when determining routes, which effects node power consumption. Also, a node may have to increase its transmission power significantly to acquire enough neighbors. The target neighbor count may not be appropriate for all nodes and determining a value for each node individually is difficult.

5.2 ECC

Studied by mathematicians for more than a century, elliptic curves claim significant coverage in the literature. Similarly has ECC received much attention since its discovery in 1985.

Woodbury recommends an optimal extension field, $\mathbb{F}_{(2^8-17)^{17}}$, for low-end, 8-bit processors[67]. In [31], Jung *et al.* propose supplementary hardware for AVR implementing operations over binary fields. In [24], Handschuh and Paillier propose cryptographic coprocessors for smart cards, whereas in [68], Woodbury *et al.* describe ECC for smart cards without coprocessors. Albeit for a different target, Hasegawa *et al.* provide in [26] a "small and fast" implementation of ECC in software over $\mathbb{F}_p$ for a 16-bit microcomputer. In [23], Guajardo *et al.* describe an implementation of ECC for the 16-bit TI MSP430x33x family of microcontrollers. Weimerskirch *et al.*, meanwhile, offer an implementation of ECC for Palm OS in [66], and Brown *et al.* offer the same in [6] for Research In Motion's RIM pager. ZigBee [70], on the other hand, shares this work's aim of wireless security for sensor networks albeit not with ECC but with AES-128.

A number of implementations of ECC in software are freely available, though none are particularly well-suited for the MICA2, in no small part due to their memory requirements. Rosing [58] offers his C-based implementation of ECC over $\mathbb{F}_{2^p}$ with both polynomial and normal bases. LibTomCrypt [14] offers another C-based implementation with a polynomial basis, as do MIRACL [41], ECC-LIB [69], and pegwit [54]. Dragongate Technologies Limited, meanwhile, offers borZoi and jBorZoi [39], implementations of ECC in C++ and Java, respectively. Another implementation in C++ is available through libecc [38].

Testament to current interest in ECC, the Workshop on Elliptic Curve Cryptography is now in its eighth year.

6 Conclusion

The MICA2 ECC and dynamic power scaling algorithms presented herein offer valuable progress toward secure, low-power sensor node communication. Our analysis of the adaptive power scheme goes beyond simulation to capture real network performance accurately in a typical indoor environment. Despite claims to the contrary, public-key infrastructure appears viable on the MICA2. Although the implementations, studied herein, of modular exponentiation and ECC in 4 KB of primary memory on this 8-bit, 7.3828-MHz device require improvement, this work's initial results are promising indeed. AVR assembly alone offers significant potential for ECC's enhancement.

The need for PKI's success on the MICA2 seems clear. TinySec's shared keys do allow for efficient, secure communications among nodes. But such devices as those in sensor networks, for which physical security is unlikely, require some mechanism for those shared keys' distribution and redistribution.

In that it offers equivalent security at lower cost to memory and bandwidth than does Diffie-Hellman based on DLP, public-key infrastructure based on elliptic curves does seem a apt choice for the MICA2. Ahead now is pursuit of this cryptosystem's minimal cost in cycles and power.

Dynamic output power scaling has the potential to maintain reliable links between nodes while saving power in some scenarios. Analysis of single link radio channel behavior and base station packet reception for node-to-sink routes showed good performance for nodes with certain connectivity characteristics. For instance, nodes close to their parent node in the routing tree were able to save several orders of magnitude of power over using the fixed, default power level. Other pairwise links did not respond well under dynamic power scaling. Radio channel quality between mid-range nodes fluctuated wildly, generating link quality oscillations to which the motes could not respond appropriately. Future work involves modifying the dynamic radio output power algorithm for better performance under these degenerate cases.

References

[1] L. M. Adleman. A subexponential algorithm for the discrete logarithm problem with applications to cryptography. In *Proc. 20th IEEE Found. Comp. Sci. Symp.*, pages 55–60, 1979.

[2] P. Barrett. Implementing the rivest shamir and adleman public key encryption algorithm on a standard digital signal processor. In A. M. Odlyzko, editor, *Advances in Cryptology – CRYPTO '86*, volume 263, 1987.

[3] G. Barwood. Elliptic curve cryptography faq v1.12 22nd. `http://www.cryptoman.com/elliptic.htm`, December 1997.

[4] E. Biham, A. Biryukov, and A. Shamir. Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. *Lecture Notes in Computer Science*, 1592:12–23, 1999.

[5] I. Blake, G. Seroussi, and N. Smart. Elliptic curves in cryptography. *LMS Lecture Note Series*, 265, 1999.

[6] M. Brown, D. Cheung, D. Hankerson, J. L. Hernandez, M. Kirkup, and A. Menezes. Pgp in constrained wireless devices. In *Proceedings of the 9th USENIX Security Symposium*. USENIX Association, August 2000.

[7] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: Application driver for wireless communications technology, 2001.

[8] Cipcon SmartRF CC1000 Datasheet. `http://www.chipcon.com/files/CC1000_Data_Sheet_2_1.pdf`.

[9] E. B. Company. Engineering datasheet: Energizer no. x91. `http://data.energizer.com/datasheets/library/primary/alkaline/energizer%/consumer_oem/e91.pdf`.

[10] Computer Security Division. *SKIPJACK and KEA Algorithm Specifications*. National Institute of Standards and Technology, May 1988.

[11] C. Corp. Remarks on the security of the elliptic curve cryptosystem. `http://www.comms.engg.susx.ac.uk/fft/crypto/EccWhite3.pdf`, July 2000.

[12] A. Corporation. *ATmega128(L) Preliminary Complete*. San Jose, CA, December 2003.

[13] MICA2: Wireless Measurement System. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/6020-0042-0%4_A_MICA2.pdf`.

[14] T. S. Denis. Libtomcrypt. `http://libtomcrypt.org/`.

[15] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[16] W. Diffie, P. C. van Oorschot, and M. J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes, and Cryptography*, 2(2):107–125, 1992.

[17] G. Frey and H. Gangl. How to disguise an elliptic curve (weil descent). ECC '98, September 1998.

[18] P. Gaudry, F. Hess, and N. P. Smart. Constructive and destructive facets of weil descent on elliptic curves. Technical Report CSTR-00-016, Department of Computer Science, University of Bristol, October 2000.

[19] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesC language: A holistic approach to networked embedded systems. In *Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation (PLDI)*, June 2003.

[20] D. M. Gordon. Discrete logarithms in gf(p) using the number field sieve. *SIAM J. Discret. Math.*, 6(1):124–138, 1993.

[21] D. M. Gordon. A survey of fast exponentiation methods. *J. Algorithms*, 27(1):129–146, 1998.

[22] J. Groβschädl. Implementation options for elliptic curve cryptography. `http://www.iaik.tugraz.at/teaching/02_it-sicherheit/04_vortragsthemen/E%CC.pdf`, April 2003.

[23] J. Guajardo, R. Blümel, U. Krieger, and C. Paar. Efficient implementation of elliptic curve cryptosystems on the ti msp430x33x family of microcontrollers. In K. Kim, editor, *PKC 2001*, pages 365–382, Korea, 2001.

[24] H. Handschuh and P. Paillier. Smart card crypto-coprocessors for public-key cryptography. In J.-J. Quisquater and B. Schneier, editors, *Lecture Notes in Computer Science*, Smart Card Research and Applications, pages 386–394. Springer-Verlag, 2000.

[25] D. Hankerson, J. L. Hernandez, and A. Menezes. Software implementation of elliptic curve cryptography over binary fields. *Lecture Notes in Computer Science*, 1965, 2001.

[26] T. Hasegawa, J. Nakajima, and M. Matsui. A Small and Fast Software Implementation of Elliptic Curve Cryptosystems over GF(p) on a 16-Bit Microcomputer. *IEICE Trans. Fundamentals*, E82-A(1):98–106, January 1999.

[27] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocols for wireless microsensor networks. In *Proceedings of the 33rd Hawaii Internation Conference on Systems Sciences (HICSS)*, January 2000.

[28] M. Hempstead, V. Shnayder, and B. rong Chen. Power TOSSIM: Efficient power simulation for TinyOS applications. CS263 final report.

[29] J. Hill, R. Szewczyk, A. Woo, S. H. D. Culler, and K. Pister. System architecture directions for networked sensors. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, November 2000.

[30] D. Johnson, D. Maltz, and J. Broch. Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks. In C. Perkins, editor, *Ad Hoc Networking*, pages 139–172. Addison-Wesley, 2001.

[31] M. Jung, M. Ernst, F. Madlener, S. Huss, and R. Blümel. A reconfigurable system on chip implementation for elliptic curve cryptography over gf(2^n). http://ece.gmu.edu/crypto/ches02/talks_files/Jung.ppt.

[32] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

[33] N. Koblitz. Cm-curves with good cryptographic properties. In *Advances in Cryptology – CRYPTO '91*, pages 279–287, 1992.

[34] V. A. Kottapalli, A. S. Kiremidjian, J. P. Lynch, E. Carryer, T. W. Kenny, K. H. Law, and Y. Lei. Two-tiered wireless sensor network architecture for structural health monitoring, March 2003.

[35] M. Kubisch, H. Karl, A. Wolisz, L. C. Zhong, and J. Rabaey. Distributed algorithms for transmission power control in wireless sensor networks. In *Proceedings of the 2003 IEEE Wireless Communications and Networking Conference (WCNC)*, March 2003.

[36] R. Laboratories. Lightweight Security for Wireless Networks of Embedded Systems. http://www.rsasecurity.com/rsalabs/pkcs/pkcs-3/, November 1993.

[37] B. A. LaMacchia and A. M. Odlyzko. Computation of discrete logarithms in prime fields. *Lecture Notes in Computer Science*, 537:616–618, 1991.

[38] libecc. http://libecc.sourceforge.net/.

[39] D. T. Limited. jborzoi 0.9. http://dragongate-technologies.com/products.html, August 2003.

[40] J. López and R. Dahab. An overview of elliptic curve cryptography. Technical report, Institute of Computing, Sate University of Campinas, São Paulo, Brazil, May 2000.

[41] S. S. Ltd. Multiprecision integer and rational arithmetic c/c++ library. http://indigo.ie/~mscott/#Elliptic.

[42] G. Marsaglia. The mother of all random generators. ftp://ftp.taygeta.com/pub/c/mother.c, October 1994.

[43] A. Menezes, S. Vanstone, and T. Okamoto. Reducing elliptic curve logarithms to logarithms in a finite field. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 80–89. ACM Press, 1991.

[44] Mica wireless measurment system. Crossbow Product Datasheet, 2003. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/6020-0041-0%1_A_MICA.pdf.

[45] Mica2 wireless measurment system. Crossbow Product Datasheet, 2003. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/6020-0042-0%4_A_MICA2.pdf.

[46] Microprocessor and M. S. Committee. *IEEE Standard Specifications for Public-Key Cryptography*. IEEE Computer Society, January 2000.

[47] V. Miller. Uses of elliptic curves in cryptography. In *Lecture Notes in Computer Science 218: Advances in Crytology - CRYPTO '85*, pages 417–426. Springer-Verlag, Berlin, 1986.

[48] P. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44(170):519–521, 1985.

[49] NEST Challenge Architecture, August 2002.

[50] N. I. of Standards and Technology. Federal Information Processing Standards Publication 185: Escrowed Encryption Standard (EES), February 1994.

[51] N. I. of Standards and Technology. Recommended elliptic curves for federal government use. http://csrc.nist.gov/CryptoToolkit/dss/ecdsa/NISTReCur.pdf, July 1999.

[52] N. I. of Standards and Technology. Special Publication 800-57: Recommendation for Key Management, January 2003.

[53] C. Paar. Implementation options for finite field arithmetic for elliptic curve cryptosystems. Invited presentation at the 3rd Workshop on Elliptic Curve Cryptography (ECC '99), November 1999.

[54] pegwit. http://groups.yahoo.com/group/pegwit/files/.

[55] C. Perkins and E. Royer. Ad hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 90–100, 1999.

[56] A. Perrig, R. Szewczyk, V. Wen, D. E. Culler, and J. D. Tygar. SPINS: security protocols for sensor networks. In *Mobile Computing and Networking*, pages 189–199, 2001.

[57] M. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, MIT, 1979.

[58] M. Rosing. *Implementing Elliptic Curve Cryptography*. Manning Publications Co., 1999.

[59] R. Schroeppel, H. Orman, S. O'Malley, and O. Spatscheck. Fast key exchange with elliptic curve systems. *Lecture Notes in Computer Science*, 963, 1995.

[60] Silverman and Suzuki. Elliptic curve discrete logarithms and the index calculus. In *ASIACRYPT: Advances in Cryptology – ASIACRYPT: International Conference on the Theory and Application of Cryptology*. LNCS, Springer-Verlag, 1998.

[61] J. Solinas. Generalized mersenne numbers, 1999.

[62] J. A. Solinas. An improved algorithm for arithmetic on a family of elliptic curves. In *Advances in Cryptology – CRYPTO '97*, pages 357–371, 1997.

[63] B. Technologies. Diffie-hellman 1, July 2003.

[64] TinySec: Link Layer Security for Tiny Devices. http://www.cs.berkeley.edu/~nks/tinysec/.

[65] Vital Dust: Wireless Sensor Networks for Emergency Medical Care. http://www.eecs.harvard.edu/~mdw/proj/vitaldust/.

[66] A. Weimerskirch, C. Paar, and S. C. Shantz. Elliptic curve cryptography on a palm os device. Sydney, Australia, July 2001. The 6th Australasian Conference on Information Security and Privacy.

[67] A. D. Woodbury. Efficient algorithms for elliptic curve cryptosystems on embedded systems. http://www.wpi.edu/Pubs/ETD/Available/etd-1001101-195321/unrestricted/w%oodbury.pdf, September 2001.

[68] A. D. Woodbury, D. V. Bailey, and C. Paar. Elliptic curve cryptography on smart cards without coprocessors. "Bristol, UK", September 2000. The Fourth Smart Card Research and Advanced Applications (CARDIS 2000) Conference.

[69] C. Zaroliagis. ECC-LIB: A Library for Elliptic Curve Cryptography. http://www.ceid.upatras.gr/faculty/zaro/software/ecc-lib/.

[70] Zigbee alliance. http://www.zigbee.org/.