The Computer Journal Advance Access published May 30, 2006

doi:10.1093/comjnl/bxl023

Security Considerations and Key Negotiation Techniques for Power Constrained Sensor Networks

Barry Doyle[1], Stuart Bell[1] and Alan F. Smeaton[2]

[1]*School of Computing and* [2]*Adaptive Information Cluster and Centre for Digital Video Processing, Dublin City University, Glasnevin, Dublin 9, Ireland*
Corresponding author: alan.smeaton@dcu.ie

Sensor networks are becoming increasingly important for a wide variety of applications including environmental monitoring, building safety and emergency relief services. A typical sensor network consists of a large number of small, low-power, low-cost nodes that form a self-organized network using wireless peer-to-peer communication. Because sensor networks pose unique constraints on their operation, traditional security techniques used by conventional networks cannot be applied. In this paper we consider the operational issues and security threats to sensor networks. We discuss the state of the art in terms of sensor network security and we examine the practicality of using efficient elliptic curve algorithms and identity based encryption to deploy a secure sensor network infrastructure. We evaluate the potential for realizing this on low-power, long-life devices by measuring power consumption of the operations needed for key management in a sensor network and thus provide further evidencefor the feasibility of the approach.

Keywords: I.2.9 Robotics:Sensors, C.2 Networks:Security, E.3 Encryption:Public Key Cryptosystems

Received 20 October 2005; revised 16 February 2006

1. INTRODUCTION

Sensor networks are attracting a lot of research interest due to their suitability in a wide variety of commercial, industrial and military uses. In military, Merrill *et al*. [1] discuss improvements in landmine technology based on energy constrained wireless communication, dynamic networking, networked sensing and intelligent command and control. The ability to self-heal and reconfigure landmines may better protect friendly military forces and could be deactivated and safely removed post-war, preventing civilian tragedies years after a conflict has ended. In health, sensor nodes can be deployed to monitor patient vital signs and assist disabled patients. In environmental monitoring the global ocean observation system is a deployed global array of sensing devices for ocean observations to help try detect and understand global climate change. Some other commercial applications include managing inventory, monitoring ocean and wildlife, monitoring disaster areas and monitoring environmental conditions.

Despite the limited computational and sensing ability of each individual sensor node, the aggregated effect of sensors working together can offer a more accurate picture of the spatial region in which the sensors are placed than conventional remote sensing technology. Sensor nodes (also referred to as 'motes') have a number of defining characteristics: they contain miniaturized sensors, limited computational ability, bi-directional wireless communications and a power supply, while being inexpensive enough to deploy by their thousands. In general, we assume that each node's location, once deployed, is fixed for its lifetime.

The remainder of this paper is organized as follows: an overview of sensor networks is provided in Section 2. In Section 3 we discuss the security issues surrounding the design, deployment and manageability of sensor nodes. We will also look at the physical, network protocol and key distribution attacks commonly used against sensor networks. In Section 4 we identify the various encryption schemes necessary to build a secure sensor network and how important power efficiency is in any proposed scheme. In light of this discussion, we present an approach to the key distribution problem using an identity-based encryption (IBE) scheme based on Tate pairings. We describe our methodology and results in Section 5. Finally, our conclusions and directions for future work are presented in Section 6.

2. SENSOR NETWORKS

An overview of wireless sensors is given in several sources: [2, 3, 4]. Typically wireless sensor nodes are small with very little RAM and ROM. They are self-powered using battery and/or solar energy and their CPUs currently operate in the MHz rather than the GHz range.

One representative sensor node is the Berkeley-designed MICA2 [5], which is a several cubic inch sensor/actuator unit with an 8 Mhz Atmel CPU, 128 kb of instruction memory, 4 kb of RAM for data, 512 kb of flash memory and a low-powered radio chip from Chipcon delivering 19.2 kbps application bandwidth up to a range of 10–20 m. At full power the MICA2 will run for about 2 weeks before exhaustion and up to 1 year using sleep modes. The embedded TinyOS operating system is capable of SKIPJACK-based encryption using its secret-key module TinySec [6] and a review of cryptography on such tiny platforms can be found in [7].

Another interesting sensor example is the SmartDust [8] concept, which was introduced by Kristofer Pister in 2001. The goal of the Smart Dust project is to build a self-contained, millimetre-scale sensing and communication platform for a massively distributed sensor network. These devices will be around the size of a grain of sand and will contain sensors, computational ability, bi-directional wireless communications and a power supply. The requirement for sensor devices to be small forfeits battery capacity since contemporary battery technology is still fairly dense and will contribute most to the device's weight. The main challenge to sensor technology is reducing energy consumption to operate in hostile or uninhabited locations for long periods of time (years rather than weeks).

Wireless sensor networks share many similarities with *ad hoc* wireless networks, including multi-hop networking and the use of media access control addresses. Several important distinctions can be made between the two however. *Ad hoc* networks cooperate to form a network without any infrastructure elements such as access points or base stations. Sensor networks have one or more points of centralized control called base stations (also known as 'sinks'), which facilitate data processing and control. An *Ad hoc* network supports routing between any pair of nodes, whereas sensor networks are focused on fault tolerant routing back to a base station. Also, neighbouring nodes in a sensor network may exhibit trust relationship behaviour using in-network event correlation, aggregation and duplicate event pruning [9]. This prevents multiple witness nodes sending the same event back to the base station and saves energy and bandwidth.

When a sensor node is deployed, it broadcasts its identity and waits for neighbouring nodes to reply. The two mutually aware nodes set their RF power at just the level needed for communication. A source-based routing protocol, based on the periodic broadcast of beacons by base stations, organizes the nodes into forests, with a base station at the root of each tree as shown in Figure 1. The beacons may also synchronize time across the nodes to facilitate power-saving modes so that nodes only use their energy-consuming RF antenna at specified times.

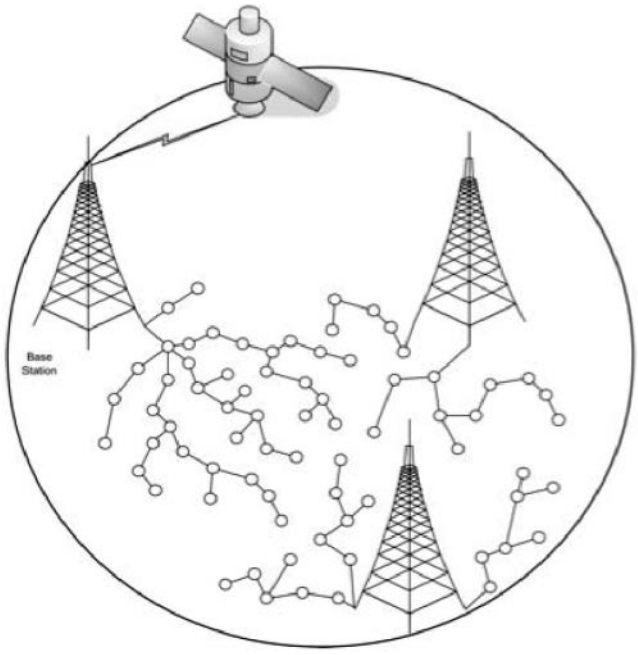

FIGURE 1. Example sensor network architecture.

The combined effects of limited wireless capacity, attenuation, multi-path propagation and interference pose a significant challenge for routing protocols operating in a sensor network. Wagner *et al*. [10] assert that they have successfully attacked all major routing protocols designed for sensor networks.

3. SECURITY ISSUES

Wireless sensor networks are vulnerable to attacks which would be more difficult to launch in the wired domain. Wireless sensor nodes are difficult to protect due to their broadcast nature, leading to situations where adversaries can easily eavesdrop, inject and alter transmitted data. In addition, adversaries are not restricted to using sensor hardware; they can interact with a sensor network from a distance by using powerful radio transceivers and workstations.

Security of the physical sensor device and underlying networking protocol is critical to the integrity of any sensor network. The following characteristics of a sensor network make for challenging problems [11].

Impracticality of public-key cryptosystems. The limited computational and power resources of contemporary sensor nodes have made it too expensive in terms of system

resources to use public-key algorithms. An RSA based Diffie–Hellman key exchange may take several minutes to complete and could also expose a node to denial of service (DoS) vulnerabilities. Recent research by Malan *et al*. [12] using an elliptic curve discrete logarithm problem (ECDLP) implementation on the MICA2 platform concluded that it took 34.161 s and used 0.816 J in order to complete a point multiplication. An important question for researchers is whether reasonable security and performance levels can be achieved with software-only cryptographic implementations or is hardware support needed?

Vulnerability of node capture. Sensor networks may be deployed in hostile locations such as public areas or battlefields. The high-volume, low-cost nature of sensor networks makes it difficult for manufacturers to produce tamper-resistant nodes. Algorithmic solutions to the problem of node capture are preferable. In the worst case, an attacker would be able to recover the cryptographic keys and pose as authorized nodes on the network.

Deployment configuration issues. The effectiveness of a sensor network may be reduced by relying too much on the base station to mediate trust relationships. If a sensor network is deployed using random scattering, the sensor nodes will not know beforehand which nodes will be neighbours. Ideally nodes should be autonomous and be able to negotiate shared keys with neighbouring nodes, independent of the base station.

Limited resources. Given the amount of memory available to each node, it will not be possible to establish a unique key with each node in the routing path. A typical sensor node also has very little bandwidth, for example the Berkeley Mica 2 platform has a transmission bandwidth of 19.2 kbps and a frame size of 30 bytes. Transmission reliability can be poor, causing expensive retransmissions especially with larger data transfers.

We now examine the physical, network and cryptographic attack surfaces of a typical sensor network.

3.1. Physical attacks

A sensor network must provide resilience against node capture and secret key recovery. Any scheme used to establish a secure infrastructure must evaluate the impact of this type of attack and the fraction of total network communication that will be compromised. Another potential threat is node replication and infiltration which may allow an attacker to populate the network with clones of the captured node and ultimately take control of the entire network. As the network grows the security of the system may be weakened and this may result in a maximum supported network size. It should also be possible to dynamically remove a detected misbehaving or hostile node.

The most common threat against an implementation of an industry standard cryptographic algorithm is the capture of a cryptographic key (either symmetric or private key). When an attacker has physical control of a sensor node she can use various techniques to extract the keying information from parts of the application-specific integrated circuit (ASIC). The attacker will focus on parts of the ASIC which are not available through normal I/O pins. For more advanced attacks, sophisticated equipment such as a focused ion beam may be used.

While there are few publications available, which accomplish physical attacks against sensor nodes, there are some related areas which could identify common threats. Gutmann [13] describes various data remanence attacks against SRAM memory cells. Of interest is the issue of encryption keys being 'burned' into memory cells over a period of time. The key values can be easily extracted even after turning off the device. Side-channel attacks also pose a significant threat for any ASIC-based system. Wollinger *et al*. [14] argue for the use of field programmable gate arrays (FPGAs) since they have no printed circuit boards and it is therefore potentially harder to find the configuration.

The low-cost nature of sensor nodes prevents the use of FIPS standard tamper-resistant modules. However some simple security-aware implementation designs may reduce some of the threats from retention effects. Gutmann [15] recommends that crypto-variables are not stored in the same location for long periods or designs which repeatedly run the same signals over dedicated data lines. He also points to the assumption made by designers that a key held in RAM is destroyed when RAM is cleared, cycling of EEPROM and flash cells 10–100 times is required prior to reuse.

3.2. Network protocol attacks

One aspect of sensor networks that complicates the design of secure routing protocols is in-network aggregation. In conventional networks, a secure routing protocol is typically only required to guarantee message availability; message integrity, authenticity and confidentiality are handled by higher levels protocols such as SSL or IPSEC. With sensor networks, in-network aggregation makes it difficult to establish end-to-end security controls because intermediate nodes may require plaintext access to the message payload.

The challenge is to build networks that operate correctly when, unknown to us, several nodes have been compromised and behave in arbitrary ways. A promising direction for building resilient networks is to replicate state across the network and to use majority voting and other techniques to detect inconsistencies. Deng *et al*. [16] have designed routing protocols that achieve some resilience against node capture by sending every packet along multiple, independent paths and checking at the destination for consistency among the packets that were received. This is at the expense of additional power consumption and bandwidth consumed.

The network attack model consists of two main classes: passive and active. A *passive* attacker does not inject data or disrupt communication; it just eavesdrops and observes activity. An *active* attacker will actively participate some or all of the time by injecting data into the device or communications stream.

Attacks on network routing protocols generally fall into two main forms of DoS attack: *routing disruption attacks* and *resource consumption attacks*.

During a routing disruption attack an attacker attempts to cause legitimate data packets to be routed in unwanted ways. During a resource consumption attack the attacker will inject packets into the network in an attempt to consume valuable network resources such as bandwidth or to consume node resources such as memory or computation cycles.

Examples of these attacks include injecting spoofed routing information, which can cause routing loops, thus consuming energy and available bandwidth. An attacker might create a routing *blackhole* which attracts and drops packets or a *grayhole* where the attacker selectively drops some packets but not others. Other effects of spoofed routing packets include creating suboptimal routes which introduce unnecessary delays or bottlenecks, or partition the network to prevent one group of nodes communicating with the base station.

A more challenging form of routing-disruption attack is a *wormhole* which can be particularly difficult to secure against. Wormhole attacks may involve two distant malicious nodes colluding to understate their distance from each other by tunnelling packets between locations in the network using an out-of-bound channel only available to the attacker. Most *ad hoc* network protocols will be unable to detect this type of attack and would result in routing table errors.

A well-placed adversary situated close to a base station may be able to completely dictate routing by creating a wormhole. If an adversary can convince nodes that they are closer to the base station then they actually are; this can create a *sinkhole*. By providing an attractive high-quality route to the base station, a wormhole can potentially attract all traffic from surrounding nodes. Figure 2 shows an example of a wormhole being used to create a sinkhole. A wormhole will work even when routing information is authenticated and encrypted; using a bridging mode, they can simply relay packets between two locations to convince the nodes they are neighbours. Hu and Perrig [17] describe a solution to this problem using *packet leases* where nodes use time-stamps or location information to determine if a packet had traversed an unrealistic distance based on the underlying network technology used.

3.3. Key distribution attacks

A major challenge is the design of protocols to establish shared secret keys between nodes which may have been pre-initialized, but have not had any prior direct contact with each other. Nodes should be able to establish secure node-to-node communication without involving the base station as an arbitrator. Additional nodes deployed at a later stage must be able to join the existing network, and unauthorized nodes must be prevented from establishing communication with the network. Various schemes have been proposed to establish privacy and authentication in a sensor network.

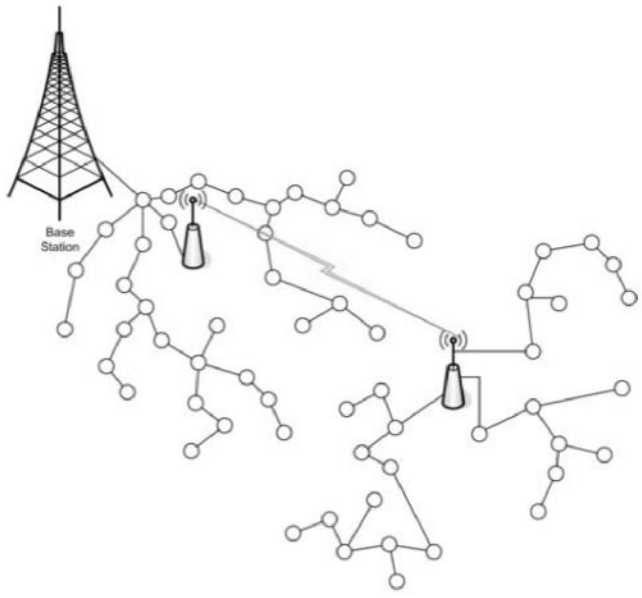

FIGURE 2. Wormhole established by adversary to create a sinkhole.

System-wide key. The simplest method is to use a shared secret key among all nodes in the network. In hostile environments the risks of node capture and key recovery are high. This intuitively rules out the use of a system-wide key since any exposure would compromise the entire sensor network.

Random key pre-distribution. Eschenauer and Gligor [18] first proposed a random key pre-distribution scheme. The scheme works as follows: prior to deployment, a large pool of keys is generated and each node is randomly populated with a small subset of keys out of the total key space. This set of keys is called a *key-ring*, chosen such that any two random subsets will share at least one key with a known probability. Once deployed, the nodes perform a broadcast-based key-discovery procedure, to find a key in common with each neighbour. Nodes can verify that their neighbours hold the key using a challenge response protocol. This key then becomes the shared key for that link.

It has been shown [18] that a network of 10,000 nodes requires a 100,000 key space and a random selection of 250 keys per node. However, this scheme is not immune to node capture and the probability of link compromise increases to 1/10 after the capture of only 40 nodes. Chan *et al*. [11] have proposed a *q*-composite scheme to increase

the resilience of the network against node capture. The operation of the q-composite key scheme is similar to that proposed by Eschenauer, differing only in the size of the key space and the use of q keys to establish communication instead of one.

While each node can verify that some of its neighbours have common keys, nodes cannot authenticate the identity of a neighbour. Since keys are issued multiple times from the same key space, other neighbouring nodes may potentially have the same shared key and could impersonate either of the communicating nodes.

Pair-wise keys. A pair-wise scheme establishes unique shared keys between two communicating nodes. This scheme supports *node-to-node* authentication which provides several key advantages. If a key k is used to create a secure link with another node, both nodes are certain of the identity of each other since no other nodes can hold k. A captured node will only contain key information about the links it has been directly involved with and there are no additional keys for the adversary to harvest. An adversary will be unable to easily impersonate a node unless it has already been captured. The scheme reduces the opportunity of node replication by increasing the key storage requirements of the adversary which of course goes against the desired characteristics of security in sensor nodes where the nodes themselves are put under very severe computing resource constraints, including memory, in order to reduce cost and make them scalable.

Much of the previous work in this area assumes a *global passive adversary*, who could monitor all communication everywhere and at all times. Anderson *et al*. [19] argue for a more realistic commercial attacker model where an attacker is only capable of monitoring a small proportion of the initial key agreement traffic. By deploying a much greater density of 'white dust' nodes than the enemies' 'black dust' nodes, pair-wise key negotiation could be conducted in plaintext or encrypted by pre-populated and disposable master key. They have demonstrated using their '*Key Infection*' protocol that it is possible to perform sufficiently secure key distribution with low computational overhead and no unnecessary key storage requirements.

Although pairwise shared keys do have advantages as outlined above, they do create a requirement for a large number of unique symmetric keys to be stored on each network node and if the size of the sensor network is large then this can place unreasonable demands on a resource-constraind network. Furthermore, because each node must have a unique key for each other node in the network, each node must verify the identity of the node that it is communicating with and this must be done through key discovery. As noted by Chan *et al*. [20] this requires some kind of challenge/response protocol which can use up unnecessary amounts of node–node communications and thus node power as not only are the nodes constrained by processing speed and memory capacity but also by the amount of power that their operation and communications consume.

Another key pre-distribution scheme was originally introduced by Du *et al*. in [21] which claimed to be scalable and flexible in that nodes do not all have to be deployed at the same time and can be added later and still be able to establish secret keys with existing nodes on the network. This was extended and further evaluated by the same authors in [22] where they also provided a framework in which to study the security of key pre-distribution schemes. This framework showed that previous approaches to handling security in key pre-distribution would be insufficient for the typical security requirements in a wireless sensor network. The framework presented in [22] will be useful to others working in this area as it will allow more direct comparisons among key distribution techniques. At around the same time as the work by Du *et al*. was published, Liu *et al*. proposed further techniques for pairwise key agreement in [23] and extended this in [24]. This presents yet two more possibilities for pairwise key pre-distribution, one based on a random subset assignment and another based on a hypercube assignment and each are examined in detail in those articles.

Each of the recent proposals for key management in wireless sensor networks [21, 22, 23, 24] has attractive features over previous proposals, notably in the area of remaining secure in the face of some proportion of the network nodes being captured or compromised, consideration for some, though not all, sensor network applications. Further trends in the area of key distribution include developing key agreement and pre-distribution techniques which are location-sensitive. This addresses the fact that in some applications network nodes can now have a level of mobility which offers both new challenges and opportunities for managing key distribution.

4. ENCRYPTION TECHNIQUES

There are many techniques for implementing security using cryptography. The security of these techniques is provided on the basis of a mathematically hard number problem. We look briefly at the two major classes of cryptographic systems in common use.

Symmetric key systems. Symmetric cryptosystems are encryption methods where the sender and receiver share the same key for encryption and decryption respectively. Symmetric key encryption schemes fall into two main categories: *stream ciphers* and *block ciphers* in various modes of operation. A stream cipher uses a *key* and an *initialization vector* (IV) as a seed to produce a large pseudo-random keystream. This keystream is then XOR'd with the plaintext message. Block ciphers work with fixed k-bit blocks of message data and keying material to produce ciphertext.

There are several modes of operation including ECB, CBC, OFB and CFB [25].

Initially, stream ciphers look attractive: we can encrypt single bytes rather than multiples of entire (possibly padded) blocks of data, also the fastest stream ciphers are generally faster than block ciphers. However stream ciphers have a critical implementation flaw: if the same IV is ever used to encrypt data with the same key it may be possible to recover the plaintexts. In a restricted environment such as a sensor node, the IV space may be too small to prevent IV reuse and stream ciphers should be avoided. Wagner *et al*. [10] recommend the use of RC5 or SKIPJACK in a variant of CBC mode on embedded processors. In a sensor network, symmetric cryptography may be used to encrypt communication once a shared secret key has been established.

Asymmetric key systems. The concept of asymmetric cryptography (also called public key cryptography) was introduced by Whitfield Diffie and Martin Hellman in 1976 in their seminal paper entitled 'New Directions in Cryptography'. Today the most popular asymmetric cryptosystems are RSA, El Gamal and elliptic curve.

The RSA cryptosystem was the first viable public key encryption algorithm; it is based on the difficulty of factoring a large integer N into two large primes (p, q) and the difficulty of finding a private decryption exponent d given only the modulus N and the public exponent e. There is no known polynomial time algorithm to factor large numbers, so it is considered a hard problem.

The elliptic curve cryptosystem (ECC) was independently introduced by Koblitz and Miller and is based on the difficulty of the discrete logarithm problem (DLP) in the group of points of an elliptic curve over a finite field. There are several standards for ECC, in particular the IEEE P1363 [26]. It is widely accepted that ECC is suited to embedded environments where memory and speed are constrained. The ECDLP provides the following hard problem: given an elliptic curve E defined over a finite field F_q, a point $P \in E(F_q)$ of order n and a point $Q \in E(F_q)$, determine the integer x, where $0 \leq x \leq n - 1$, such that $Q = xP$, provided that such an integer exists.

Asymmetric key systems are typically more expensive operations on sensors devices and are mainly used to negotiate shared secrets with other participating nodes. For all of these cryptosystems, key length is an important consideration for the security of a system. The key length used by a cryptographic protocol determines the highest security it can offer. NIST [27] have published a comparative table of equivalent security levels for symmetric and asymmetric systems.

As Table 1 shows, an ECC algorithm using a 160 bit key offers the same level of security as an RSA algorithm using a 1024 bit key.

TABLE 1. Equivalent key sizes for symmetric, ECC and RSA

Sym key (bits)	ECC key (bits)	RSA key (bits)
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	512	15,360

4.1. Elliptic curve cryptography

A non-supersingular elliptic curve E over GF(2^m) is efficient for computer computation, as each element of the field can be treated as an m-bit string. It is defined by the parameters $a, b \in$ GF(2^m) satisfying $b \neq 0$ and consists of a *finite* set of points, $P = (x,y)$ where $x, y \in$ GF(2^m) and the point at infinity θ are solutions to the equation:

$$y^2 + xy = x^3 + ax^2 + b$$

Hyperelliptic curve cryptography (HECC) is similar to elliptic curve cryptography where an algebraic geometry construct of a hyperelliptic curve with an appropriate group law provides an Abelian group on which to do arithmetic. The use of hyperelliptic curves in cryptography was first suggested by Neal Koblitz [28]. A hyperelliptic curve is an algebraic curve given by an equation of the form

$$y^2 = f(x)$$

where $f(x)$ is a polynomial of degree $n > 4$ with n distinct roots. Because the arithmetic on hyperelliptic curves is more complicated than that on elliptic curves, a properly implemented cryptosystem based on hyperelliptic curves can be more secure than elliptic curve based cryptosystems for the same key size. It is widely accepted that for most cryptographic applications based on EC or HEC the minimum group order size is $\approx 2^{160}$. With HECC over F_q we will need at least $g \cdot \log_2 q \approx 2^{160}$, where g is the *genus* of the curve. Therefore, we would need a field order size $q \approx 2^{40}$ for a genus 4 HEC. In this case, a 40 bit HEC operand would be comparable to a 160 bit ECC operand or a 1024 bit RSA operand in order to achieve the same security.

There has been a tension between the issue of computational overhead on resource-constrained sensor network nodes on the one hand, and the security demands that some sensor network applications place on the other. For example, there can be a requirement that the network remains secure even when a high proportion of the nodes have been captured or are compromised. Arazi *et al*. point out in [29] that recent work has challenged the previous assumption that public-key-based schemes are infeasible in wireless sensor networks and that we can now demonstrate that elliptic curve cryptographic key generation can be run on resource-constrained network nodes which thus yields high security

even for small sized keys. One interesting implementation approach is to use a mixture of larger and smaller keys for increased computational speed and reduced power consumption. It is also expected that the developers of sensor nodes will start to incorporate security primitives within the CPU instruction sets embedded in next generation nodes and this will offer even more efficient execution of core ECC operations executed directly in standard hardware.

We will now look at an encryption scheme which uses these important features of ECC and HECC to establish secure communication and key agreement.

4.2. Identity-based encryption

An IBE [30, 31] scheme is a public-key cryptosystem where any string may be used as a valid public key. IBE was first proposed by Adi Shamir in 1984 to help the deployment of public key infrastructures. More generally, IBE can simplify systems that manage a large number of public keys. Modern schemes include Boneh and Franklin's [32] pairing-based encryption scheme, and Cocks's encryption scheme based on quadratic residues.

In the context of sensor networks, each node's unique identification ID could be used as the public key. A trusted service in the form of a private key generator (PKG) is required to pre-populate each node with their private key and system-wide parametersprior to deployment. One of the major advantages of any IBE scheme is that if there are only a finite number of nodes, after all nodes have been issued with keys, the PKG's master secret can be destroyed. This can take place because this system assumes that, once issued, keys are always valid (as this basic system lacks any method of key revocation).

The most efficient IBE schemes are currently based on bilinear [33] pairings on elliptic curves, such as the Weil and Tate pairings.

4.3. Tate pairings

The Tate pairing, $\hat{e}$, is a bilinear non-degenerate map [33] from two points, $P \in E(\mathrm{GF}(2^m))$ and $Q \in E(\mathrm{GF}(2^{mk}))$ to an element $g \in \mathrm{GF}(2^{mk})$ such that:

$$\hat{e}_l(aP, bQ) = \hat{e}_l(aP, Q)^b = \hat{e}_l(P, Q)^{ab} = g$$

where a and b are integers, the point P is of prime order l and the point Q is a representative member of one of the cosets. In practice, l is chosen so that it divides mk and should be at least the order of 2^{160} to be secure.

The Tate pairing can be computed using an algorithm first proposed by Miller [34, 35]. Miller's algorithm is the same 'double and add' algorithm for elliptic curve point multiplication combined with an evaluation of certain intermediate functions.

A process which uses the Boneh–Franklin IBE scheme [32] in a sensor network is now described. The PKG picks a random integer s, a base point P and an elliptic curve of suitable form. It places the point P and sP on all nodes. It uses ahash function $h_1(ID)$ to map the node ID to a point Q on the curve. This value is multiplied by s with result sQ, which is now treated as the node's private key and this is placed on the node.

In order for *Node A* to encrypt a message, $m \in \{0, 1\}^n$, to *Node B*, *A* performs the following Algorithm.

Encryption
$Q = h_1(i.d.)$
random $\sigma \in \{0, 1\}^n$
integer $r = h_3(\sigma, m)$
$g \in \mathrm{GF}(2^{mk}) : g = \hat{e}_l(Q, sP)^r$
$V = \sigma \oplus h_2(g)$
$W = m \oplus h_4(\sigma)$
$U = r'P$
Transmits(V, W, U)

Note that Q and $\hat{e}_l(Q, sP)$ will only have to be calculated once and can be stored in persistent memory for future encryption. The cryptographic hash functions (h_1, h_2, h_3, h_4) are defined in [32].

In order for *Node B* to decrypt the message, (V, W, U), which it has received from *Node A*, *B* performs the following Algorithm. *B*'s private key is sQ which was placed on the device by the PKG when the node was programmed.

Decryption
$g \in \mathrm{GF}(2^{mk}) : g = \hat{e}_l(sQ, U)$
$\sigma = V \oplus h_2(g)$
$m = W \oplus h_4(\sigma)$
$r' = r = h_3(\sigma, m)$
if $U \neq r'P$ **then**
reject encrypted data
end if

where r' is a non-adjacent form (NAF) integer.

Since any node's private key sQ can be generated by the *PKG*, this system has inherent key escrow.

5. IMPLEMENTATION

So far in this paper we have presented a survey and an analysis of the major issues in security in wireless sensor networks. Our specific contribution to this rapidly developing area is to analyse the feasibility of using a Tate pairing-based approach to solve the key distribution problem.

There are several feasibility aspects to consider including whether the network nodes have the computational capacity or even the RAM to complete the calculations in reasonable

time. We are specifically interested in examining the power requirements for Tate pairing and in calculating whether Tate pairing methods would consume too much energy to make it feasible. For that reason we decided not to measure the execution time for any particular operation as this is less important than the power consumption because operations like Tate pairing and key exchange are run only a small number of times at network setup or when a node joins an existing network. For other operations on a sensor network node which are run more regularly, like actual sensing or data processing, execution time for operations would be important to know.

In a conventional mote-based sensor node one of the main power draws comes from RF-based communications between nodes. Typically a mote device will use between 10 and 30 mA (1 mW power/−10 dBm) when transmitting data over an RF link, with the current directly related to the output power of the radio and hence the communication range. When considering key exchange as part of a security infrastructure, it is thus desirable to reduce the amount of inter-node communications needed by reducing the number and size of packets needed. Tate pairing is highly suited to these restrictions since there is no need for a handshaking exchange to take place between nodes prior to sending encrypted data. The actual measurement of power consumption required for communications overhead is not something we measured and further details of how this can be done can be found in [36, 37].

In order to complete our experiments, we needed a target sensor platform, code to implement the Tate pairing and hardware or software to measure the power consumption of the Tate pairing code. We selected an ARM RISC processor as a representative target choice because we are already actively researching power analysis on the ARM architecture and have access to a variety of tools to measure energy consumption on this platform. Before we discuss the tools we used, we will briefly introduce some energy profiling concepts.

There are two main approaches to acquiring energy consumption information: simulation and measurement. Using simulation-based energy estimations is convenient in software development environments, however they may not provide enough accuracy as suggested in [38]. A measurement-based approach tends to be more accurate provided its possible to sample the high-frequency energy spikes inherent in digital systems. With software size increasing, research into power-aware compilers is seeking ways to produce machine code with optimized energy consumption. A power-aware compiler will typically consider the energy requirements of one instruction sequence from a set of alternatives, and in systems with more than one type of memory (e.g. on-chip and off-chip) the compiler must decide on the location of memory objects, and also the impact of instruction scheduling and state changes.

5.1. Research tools used

We briefly describe the selected hardware and software tools used in this research.

ARM processor. The target processor chosen for this paper is the ARM7DTMI [39] test chip, which is optimized for low power environments. The ARM7TDMI processor employs a unique architectural strategy known as THUMB [40]. THUMB provides a super-reduced 16-bit instruction set which allows it to approach twice the density of standard ARM code.

ARM RealView Development Suite (*RVDS*). The RealView Developer Suite provides tools for building, debugging, and managing software development projects targeting ARM processors. It includes a version of Metrowerks CodeWarrior IDE version 5.6 specifically for RVDS. Critically, the RealView Debugger provides support for tracing with either trace hardware or a hardware simulator provided by the RealView ARMulator. Trace files provide the necessary instruction level detail required to approximate power consumption of executed code.

MIRACL library. MIRACL [41] is a large number C and C++ Library which implements all of the primitives necessary to implement cryptographic code. It also offers full support for Elliptic curve cryptography over GF(p) and GF(2^m), which we require for Tate pairing.

Enprofiler. Enprofiler [42] provides statistical information about the execution of programs on the ARM7DTMI processor. The statistics generated consist of the program's energy dissipation, considering the consumption of each individual instruction. From this data (including inter-instruction effects), performance statistics are generated which also contain the number of executed instructions, number of cycles and number and energy consumption of memory accesses (considering different types of memory).

Tate pairing code. Baretto *et al*. [43] have implemented Tate pairing based on Duursma and Lee for a special family of supersingular hyperelliptic curves. They suggest that their techniques may provide similar efficiencies and performance to elliptic curves. In [44], they present a general technique for the efficient computation of pairings on supersingular Abelian varieties. This formulation, which they call the *ETA pairing*, generalizes the results of Duursma and Lee and provides improvements to the Miller algorithm which are approximately twice as fast as the Duursma–Lee method.

5.2. Methodology

We compiled the MIRACL source code using RVDS into a THUMB image library. This library was included in the build of executable images to benchmark elliptic curve point manipulation and implement the two Tate pairing methods from [43] and [44]. Obviously this led to an executable

image size which was not optimized as it contained the MIRACL library and our small C program, and thus would not be suitable for a sensor node. Our next steps in an implementation would be to strip down the MIRACL library to the bare minimum required and optimized hardware design, possibly using an FPGA as used in the Tyndall mote sensor platform [45], would make this efficient. In our implementation we used the RDVS Debugger to simulate an ARM7DTMI processor and traced the execution of the images to file. Using enprofiler we measured the energy consumed from the trace file contents.

5.3. Results

To facilitate the collection of trace information, we were restricted to curves over GF(2^{107}). We note that this curve would not provide a secure system since the minimium key size for characteristic two curves is 2^{283}, as recommended by NIST. Using point operations for our initial benchmark calculation, we measured the execution time and power consumption for points in a curve. Our ARM simulator was configured to run at 50 Mhz.

As Table 2 shows, the energy difference between a single point doubling and doubling five times is $\approx$1.2 mJ. This indicates that the largest portion of energy is attributed to program initialization including curve setup and memory allocation. By subtracting a single point doubling from the five point doubling results, we can extrapolate that a single (zero-overhead) point doubling cost is $\approx$0.31 mJ. For point addition we added five distinct points on the curve and similarly for subtraction, we subtracted five points. The difference between adding and subtracting using these five points was $\approx$1.3 mJ. When we compare the energy results to [12] of 0.816 J for scalar multiplication, we attribute the difference in energy to the selection of the elliptic curve over GF(2^{163}) versus our curve in GF(2^{107}).

We present our calculations of the energy consumption needed to implement Tate pairing, based on [44] on an ARM7DTMI processor in Table 3. As Table 3 shows there is a considerable improvement in the energy consumption performance between the original Duursma–Lee algorithm and the *ETA pairing* version. This equates to a 18.5% saving in energy consumed and goes some way to validating our hypothesis that Tate pairing can be used for key management in a power-constrained wireless sensor network.

TABLE 2. Energy Consumption and execution time based on point operations on a supersingular elliptic curve in GF(2^{107}).

Operation	Energy (mJ)	Time (s)
Point doubling	216.73	1.350
Point doubling $\times$ 5	217.95	1.526
Point addition	256.70	1.339
Point subtraction	255.42	1.327

TABLE 3. Energy consumption based on calculation of a Tate pair using algorithms from [43] and [44].

Algorithm	Power Consumption (J)
Duursma–Lee	0.545
Faster Duursma–Lee	0.444

6. CONCLUSIONS AND FUTURE WORK

The severe constraints and demanding deployment environments of wireless sensor networks make securing these systems a challenging problem. In this paper we have examined the physical, networking and key management issues of sensor networks and various attacks that can be launched against them. We then focused on the key distribution problem and described how an IBE scheme could be used to negotiate a shared secret key which is then used by the nodes to symmetrically encrypt node–node communications. In order to realise this in practice, however, we require calculation of pairs of keys to be done in a way which is not unreasonably demanding in terms of time or power consumption. We examined the energy performance of key negotiation using an IBE scheme based on Tate pairing on the ARM platform. Our results indicate that a single Tate pairing using the *ETA* approach consumed 0.44 J, which we believe is an acceptable cost for the infrequent distribution of secret keys between nodes where nodes may be limited to a 1000 J battery capacity.

Many aspects of this energy analysis require further research. Specifically, our results should be compared with actual measurements on ARM trace hardware. Furthermore, additional energy aspects could be considered within enprofiler such as the impact of 'cache memory' and 'additional wait states' on the energy performance.

ACKNOWLEDGEMENTS

The authors wish to acknowledge assistance from Kealan McCusker, Andrew Kinane and Alan Casey. We also would like to thank Lars Wehmeyer for his help with *enprofiler*, and Mike Scott for use of MIRACL. This work was partly funded by Science Foundation Ireland under grant number 03/IN.3/I361.

REFERENCES

[1] Merrill, W. M., Girod, L., Schiffer, B., McIntire, D., Rava, G., Sohrabi, K., Newberg, F., Elsonand, J. and Kaiser, W. (2004) Dynamic networking and smart sensing enable next generation landmines. *IEEE Pervas. Comput.*, **3**, 84–90.

[2] Wikipedia (2006) Sensor networks. Available at http://en.wikipedia.org/wiki/Sensor_Networks.

[3] Akyildiz, I., Su, W., Sankarasubramaniam, Y. and Cayirci, E. (2002) Wireless sensor networks: a survey. *Comput. Netw.*, **38**, 393–422.

[4] Perrig, A., Stankovic, J. and Wagner, D. (2004) Security in wireless sensor networks. *Commun. ACM*, **47**, 53–57.

[5] Technology Inc., C. (2006) Motes, smart dust sensors, wireless sensor networks. Available at: http://www.xbow.com/Products/Wireless_Sensor_Networks.htm.

[6] Karlof, C., Sastry, N. and Wagner, D. (2004) Tinysec: a link layer security architecture for wireless sensor networks. In *Proc. Second ACM Conf. Embedded Networked Sensor Systems (SenSys 2004)*, Baltimore, MD, USA, November, pp. 69–80. ACM Press.

[7] Malan, D. J. (2004) Crypto for tiny objects. TR-04-04, Computer Science Group, Harvard University.

[8] Warneke, B., Last, M., Liebowitz, B. and Pister, K. S. J. (2001) Smart dust: communicating with a cubic-millimeter computer. *Computer*, **34**, 44–51.

[9] Karlof, C. and Wagner, D. (2003) Secure routing in wireless sensor networks: attacks and countermeasures. In *Proc. First IEEE Int. Workshop on Sensor Network Protocols and Applications*, May, pp. 113–127. IEEE.

[10] Karlof, C., Sastry, N. and Wagner, D. (2004) Tinysec: a link layer security architecture for wireless sensor networks. In *Proc. 2nd Int. Conf. Embedded Networked Sensor Systems, SenSys '04*, Baltimore, MD, USA, pp. 162–175. ACM Press.

[11] Chan, H., Perrig, A. and Song, D. (2003) Random key predistribution schemes for sensor networks. In *Proc. 2003 IEEE Symp. Security and Privacy, SP '03*, Washington, DC, USA, pp. 197. IEEE Computer Society.

[12] Malan, D. J., Welsh, M. and Smith, M. D. (2004) A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography. In *Proc. First IEEE Int. Conf. Sensor and Ad Hoc Communications and Networks*, Santa Clara, CA, October. IEEE.

[13] Gutmann, P. (1996) Secure deletion of data from magnetic and solid-state memory. In *Proc. 6th USENIX Security Symp.*, San Jose, CA, July. USENIX.

[14] Wollinger, T., Guajardo, J. and Paar, C. (2004) Security on FPGAS: Scdvplex-of-the-art implementations and attacks. *Trans. Embed. Comput. Syst.*, **3**, 534–574.

[15] Gutmann, P. (2001) Data remanence in semiconductor devices. In *Proc. 10th USENIX Security Symp.*, Washington, DC, pp. 39–54.

[16] Deng, J., Han, R. and Mishra, S. (2003) A performance evaluation of intrusion tolerant routing in wireless sensor networks. In *Proc. 2nd IEEE Int. Workshop on Information Processing in Sensor Networks*, April, pp. 349–364. IEEE.

[17] Hu, Y.-C. and Perrig, A. (2004) A survey of secure wireless *ad hoc* routing. *IEEE Secur. Privacy*, **2**, 28–39.

[18] Eschenauer, L. and Gligor, V. (2002) A key-management scheme for distributed sensor networks. In *Proc. 9th ACM Conf. Computer and Communications Security*, Washington, DC, November, pp. 41–47.

[19] Anderson, R. J., Chan, H. and Perrig, A. (2004) Key infection: smart trust for smart dust. In *Proc. 12th IEEE Int. Conf. Network Protocols*, Berlin, Germany, October, pp. 206–215. IEEE Computer Society.

[20] Chan, H., Perrig, A. and Song, D. (2004) Key distribution techniques for sensor networks. In Znati, T. and Sivalingam, K. (eds), *Wireless Sensor Networks*. Kluwer Academic Publishers, Dordtrecht.

[21] Du, W., Deng, J., Han, Y. S. and Varshney, P. (2003) A pairwise key pre-distribution scheme for wireless sensor networks. In *Proc. 10th ACM Conf. Computer and Communications Security (CCS'03)*, Washington, DC, October.

[22] Du, W., Deng, J., Han, Y. S., Varshney, P. K., Katz, J. and Khalili, A. (2005) A pairwise key predistribution scheme for wireless sensor networks. *ACM Trans. Inform. Syst. Secur.*, **8**, 228–258.

[23] Liu, D. and Ning, P. (2003) Establishing pairwise keys in distributed sensor networks. In *Proc. 10th ACM Conf. Computer and Communications Security (CCS '03)*, Washington, DC, October.

[24] Liu, D., Ning, P. and Li, R. (2005) Establishing pairwise keys in distributed sensor networks. *ACM Trans. Inform. Syst. Secur.*, **8**, 41–77.

[25] Smart, N. (2004) *Cryptography: An Introduction*. McGraw-Hill.

[26] (2005). The IEEE P1363 home page: standard specifications for public-key cryptography. Available at http://grouper.ieee.org/groups/1363/. IEEE.

[27] (2001). Key management guideline—workshop document, draft. Available at http://csrc.nist.gov/encryption/kms/key-management-guideline-(workshop)%.pdf. NIST.

[28] Koblitz, N. (1990) A family of Jacobians suitable for discrete log cryptosystems. In *Proc. Advances in Cryptology, CRYPTO '88*, Santa Barbara, CA, USA, pp. 94–99. Springer-Verlag, New York.

[29] Arazi, B., Elhanany, I., Arazi, O. and Qi, H. (2005) Revisiting public-key cryptography for wireless sensor networks. *IEEE Comput.*, **38**, 103–105.

[30] Boneh, D. and Franklin, M. (2001) Identity-based encryption from the weil pairing. In Kilian, J. (ed.), *Proc. Advances in Cryptology—CRYPTO 2001: 21st Annual Int. Cryptology Conf.*, Santa Barbara, CA, USA, August, pp. 213–. Springer, .

[31] Boneh, D. and Franklin, M. (2003) Identity-based encryption from the Weil Pairing. *SIAM J. Comput.*, **32**, 586–615.

[32] Boneh, D. and Franklin, M. (2003) Identity based encryption from the Weil pairing. *SIAM J. Computing.*, **32**, 586–615.

[33] Dutta, R., Barua, R. and Sarkar, P. (2004) Pairing-based cryptographic protocols: a survey. Cryptology ePrint Archive, Report i2004/064. Available at http://eprint.iacr.org/.

[34] Miller, V. S. (2004) The Weil pairing, and its efficient calculation. *J. Cryptol.*, **17**, 235–261.

[35] Frey, G., Miller, M. and Ruck, H.-G. (1999) The Tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. *IEEE Trans. Inform. Theory*, **45**, 1717–1719.

[36] Pottie, G. J. and Kaiser, W. J. (2000) Wireless integrated network sensors. *Commun. ACM*, **43**, 51–58.

[37] Pollin, S., Bougard, B., Mangharam, R., Van der Perre, L., Catthoor, F., Rajkumar, R. and Moerman, I. (2005) Optimizing transmission and shutdown for energy-efficient packet scheduling in sensor networks. In *Proc. Second European Workshop on Wireless Sensor Networks*, Istanbul, Turkey, pp. 290–301. IEEE.

[38] Steinke, S., Knauer, M., Wehmeyer, L. and Marwedel, P. (2001) An accurate and fine grain instruction-level energy model supporting software optimizations. In *Proc. Int. Workshop on Power And Timing Modeling, Optimization and Simulation (PATMOS)*, September 2001.

[39] RISC Machines (ARM) Ltd, A. (2006) ARM7TDMI ARM 32-bit RISC core with 16-bit system costs. Available at http://www.arm.com/products/CPUs/ARM7TDMI.html.

[40] Machines (ARM) Ltd, A. R. (1995) ARM 7TDMI Datasheet. Document Number: ARM DDI 0029E.

[41] Scott, M. (2005) Multiprecision integer and rational arithmetic cc++ library. Shamus Software Ltd. Available at http://indigo.ie/~mscott/.

[42] Steinke, S., Knauer, M., Wehmeyer, L. and Marwedel, P. (2001) An accurate and fine grain instruction-leel energy model supporting software optimizations. *Int. IEEE Workshop on Power and Timing Modeling, Optimization and Simulation, PATMOS 2001*, Yverdon-les-bains, Switzerland, September. IEEE.

[43] Barreto, P. S. L. M., Lynn, B. and Scott, M. (2004) Efficient implementation of pairing-based cryptosystems. *J. Cryptol.*, **17**, 321–334.

[44] Barreto, P. S. L. M., Galbraith, S., O'hEigeartaigh, C. and Scott, M. (2004) Efficient pairing computation on supersingular Abelian varieties. Cryptology ePrint Archive, Report 2004/375. Available at http://eprint.iacr.org/.

[45] O'Flynn, B., Bellis, S., Delaney, K., Barton, J., O'Mathuna, S., Barroso, A., Benson, J., Roedig, U. and Sreenan, C. (2005) The development of a novel minaturized modular platform for wireless sensor networks. In *Fourth Int. Symp. Information Processing in Sensor Networks*, Los Angeles, CA, USA, April, pp. 370–375. IEEE.