DHB-KEY: An Efficient Key Distribution Scheme for Wireless Sensor Networks

Tony Chung and Utz Roedig
Email: {{a.chung|u.roedig}@lancaster.ac.uk}
Infolab21, Lancaster University, UK

***Abstract*—Real-world deployments of wireless sensor networks require secure communication. In many application cases it is sufficient to provide message authentication at the sink. To implement this requirement using symmetric ciphers, keys shared between each sensor node and the sink have to be established and kept fresh during network operation. This paper presents a key distribution scheme based on the well known Elliptic Curve Diffie-Hellman key exchange mechanism that allows us to fulfil the previously outlined requirements efficiently. The DHB-KEY scheme requires only the distribution of a single sink-initiated broadcast message to set individual keys on all sensor nodes. Thus, DHB-KEY has a low complexity and preserves scarce resources such as bandwidth and energy. In the paper we present a protocol specification based on the DHB-KEY scheme and its implementation for the well known TinyOS platform. A physical intrusion detection system in an office building is used to evaluate the protocol implementation. The evaluation shows that DHB-KEY is practical in real-world deployments.**

I. Introduction and Motivation

Some sensor network applications require secure communication. In many cases it is sufficient to secure the end-to-end data transport between the sensor nodes and the sink used for data analysis. In particular, the sink must be able to verify that the received data was generated by a specific node and not modified in transit; in rare cases, data confidentiality is required as well. To implement the necessary cryptographic methods, such as Message Authentication Codes (MAC), keys have to be negotiated between each node and the sink.

A unique key should be used for each sensor node to facilitate a simple key management mechanism. Key revocation might be necessary in case that a node is deemed to have become untrustworthy. Maintenance procedures might require node replacement or addition. The use of symmetric keys is desirable as symmetric cryptographic algorithms require lesser computational effort compared to public key cryptography on the resource constrained sensor nodes. To make cryptanalysis infeasible it is desirable to refresh keys regularly.

A key distribution mechanism has to take the specific WSN communication properties into account. Obviously, the key distribution mechanism should require as few messages as possible as the distribution of messages is energy costly. However, energy consumption is not the sole reason for reducing message numbers. Available network capacity should be available to transport sensor data and not be (temporarily) consumed by key distribution messages. For example, if time critical sensor data has to be transported to the sink the network should not be congested at that time with key distribution messages. Besides the necessary message number, the network structure needs to be taken into account when constructing a key distribution mechanism. Most sensor networks are optimised for data transport from the sensor nodes to the sink and consequently most network resources are allocated for this traffic direction. Thus, it is not feasible to construct a key distribution protocol that requires bi-directional traffic flow between the sink and all sensor nodes. Finally, a high number of packet losses can be observed in wireless sensor networks. Therefore, any key distribution mechanism must be able to deal with frequent losses and it must be possible to integrate recovery mechanisms that are not too resource hungry.

Existing key distribution mechanisms proposed for wireless sensor networks do not match the outlined requirements. Often keys are negotiated such that all nodes can securely communicate with all other nodes in the network [5]. We believe that for many scenarios this is not required and introduces unnecessary complexity. Other solutions require the exchange of many messages between the sink and each node to negotiate keys [4]. Thus, a large portion of available resources in the network have to be spent on key distribution rather than application related tasks. Most importantly, existing key distribution mechanisms are not aligned with network properties observed in WSNs.

To overcome the outlined limitations, we propose DHB-KEY (outlined in [1]), a key distribution mechanism for wireless sensor networks based on Elliptic Curve Diffie-Hellman. The first part of DHB-KEY is conducted before network deployment. The second part is initiated periodically by the sink using a broadcast message. As the first part is executed in a secure environment, the man in the middle problem common to Diffie-Hellman scenarios is avoided. The use of a broadcast message to complete the second part allows

978-1-4244-2575-4/08/$20.00

us to minimise the communication overhead required to perform a key exchange and to align the key exchange protocol with the present network structure.

Our paper is organised as follows. In the next section we discuss related work. Section III introduces a WSN application scenario that benefits from the introduced DHB-KEY mechanism and is used for evaluation in later sections. Section IV details the DHB-KEY mechanism, constituting the first contribution of this paper. Section V details a protocol specification using DHB-KEY and its implementation in TinyOS as a second contribution. In Section VI we evaluate DHB-KEY analytically and by studying a real-world WSN deployment. We finish with a conclusion in Section VII.

II. Related Work

Many existing security schemes work at the link-layer, securing communication only between adjacent nodes (for example, TinySec [3]). End-to-end security thus relies on the security of multiple nodes along the communication path. The main weakness is that a compromised node can modify all messages that pass through without detection. Also, a node's key cannot be revoked without potentially disconnecting a large portion of the network, further complicating the problem. Many cluster-based schemes are similarly limited (for example, [12]) as they distribute authentication in a similar manner. Some deployments further generalise by using a single network key, leaving the entire network vulnerable if it is compromised [5].

It would be desirable to use public key algorithms, such as DSA, to provide end-to-end authentication directly. Unfortunately the restricted computational capability and power supply of wireless sensor nodes is insufficient to support this [8], [3]. Many existing methods therefore rely on full symmetric schemes (for example, [8], [11], [12]). These solutions exhibit either the insecurity outlined above or a high communication overhead as keys are negotiated.

A simple method to establish keys on each node is to do so at deployment [8]. Although this can safely establish the keys, network maintenance becomes difficult and keys cannot be easily and securely refreshed during deployment. Some schemes are intended for peer-to-peer communication security do not extend to end-to-end security or do so inefficiently. For example, probabilistic key sharing [11] can be used to secure communication between arbitrary nodes in the network but is not useful to support scenarios in which there is a common end-point (the sink). Multipath key establishment [4] is another method used to securely establish keys by sending portions over redundant pathways, thus requiring an attacker to compromise multiple nodes to recover the key. Although this scheme does work for end-to-end security, it requires that the topology supports fully redundant pathways, which is complex, and requires greater communication overhead.

Use of the Diffie-Hellman protocol in sensor networks has recently become feasible (see [10], [2]). However, each node still has to negotiate keys individually with the sink. As sensor networks need to be conservative with communication and are generally optimised for node-to-sink (not sink-to-node) communication, this is also undesirable.

As shown, currently available key distribution schemes for sensor networks are limited to specific scenarios, are too complex or do not provide the required level of security. Thus, many sensor network applications are insecure because designers are compelled to use inadequate or no security. The key distribution approach proposed in this paper provides keys for end-to-end security, is simple to execute and fits the communication patterns observed in sensor networks.

III. Application Scenario

We are experimenting with a physical intrusion detection system used to secure an office building. A number of tamper resistant[1] wireless sensors report their observations to a sink for data analysis. In addition, each node sends periodic heart-beat messages to indicate correct operation. The sink generates an alarm if an intrusion or system failure is detected. The system has to distinguish two alarm types:

- *Intrusion Alarm:* The sink receives an intrusion detection message. This alarm indicates a *definite breach of security*[2]. The location of the intruder is known.
- *Failure Alarm:* The sink does not receive a heart-beat message. This alarm indicates a *possible breach of security* as the alarm could be triggered by an attacker or have a non-security related cause. The exact location of the problem is unknown.

If an attacker needs at least a few minutes to leave a building after entering a restricted area, a delay of many seconds before reacting to a Failure alarm is reasonable. Our experiments (see Section VI) show that a wireless sensor system in a building experiences a fluctuating link quality and message losses occur frequently. The delay allows us to avoid reacting to failure alarms caused by a temporarily unavailable communication link. The

[1] We assume that an attacker has only the ability to destroy a node but not to tamper with it. In practice, security sensors monitor their own security and are fitted with tamper alarms which disable the node when triggered. For example, it can be arranged that the area monitored by an infrared detector has to be crossed to reach the detector. A door contact can be fitted such that the door has to be opened to reach the sensor. Opening the case of a sensor node will erase its program and render the node useless.

[2] Under the assumption that detectors have no detection errors.

network might recover during the delay period and the failure alarm can be cancelled.

The message transport in the intrusion detection system has to be secured cryptographically. In particular, message authentication at the sink is required. The sink must be able to verify that a message was created by a trusted sensor node. Message confidentiality is not a major concern as the content of messages is obvious (messages contain sensor readings). An attacker will be able to inject messages in the network as only the sink is able to verify the message authentication code. If nodes would silently discard injected messages in the network we would not know about the attacker trying to manipulate the system. Complete flooding of the network by an attacker can be prevented by limiting the forwarding rate of nodes.

To reduce the computational effort to secure a message symmetric algorithms should be used. Hence, it is necessary to establish a shared key between each node and the sink. This key has to be replaced regularly to avoid cryptanalysis. Thus, an efficient key exchange mechanism is required.

Messages in the physical intrusion detection system are flowing from all sensors towards the sink. Hence, the network should be optimised to support efficient data transport from nodes to the sink. The only data required to travel from the sink to sensor nodes are messages to update key material on the nodes. The number of these messages should be reduced such that bandwidth is available to transport messages from the sensors to the sink. In particular, it has to be guaranteed that intrusion messages from a sensor can travel quickly to the sink.

The DHB-KEY distribution mechanism can be used to support efficiently the outlined application of a physical intrusion detection system for buildings. We implemented this application scenario and used its deployment to evaluate the proposed DHB-KEY key distribution mechanism.

IV. Diffie-Hellman Broadcast Key exchange

This section describes the *D*iffie-*H*ellman[3] *B*roadcast (DHB) Key exchange mechanism. First, the well known Diffie-Hellman (DH) method based on elliptic curve cryptography is briefly summarised to define the syntax of DH parameters in the context of this paper. Second, the DHB-Key mechanism as modification of the standard DH-Key exchange is outlined. Finally, the security of the DHB-Key mechanism is discussed.

[3]The main characteristic of DHB-KEY is the use of broadcast. Other algorithms, for example using hashes, may also exhibit compatible behaviour but may not offer the security of a public key algorithm such as Diffie-Hellman. For example, Diffie-Hellman offers greater security in the event of sink compromise or node private key replacement.

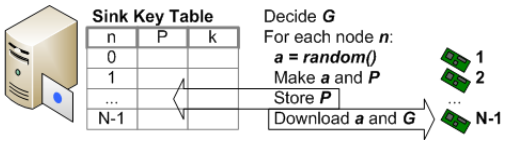

Figure 1. DHB-Key phase 1.

A. Elliptic Curve Diffie-Hellman

The Elliptic Curve variation of Diffie-Hellman (ECDH) provides similar security to normal Diffie-Hellman with significantly shorter keys. Within ECDH [2], a secret k between parties A and B is established. This secret k may then be converted into cryptographic keying material.

1) A and B agree ECDH parameters with curve base G.
2) A generates private number a and public point $P = Ga$.
3) B generates private number b and public point $Q = Gb$.
4) P and Q are exchanged over an insecure channel.
5) A generates a secret $k_a = Qa$.
6) B generates a secret $k_b = Pb$.
7) The shared secret is: $k = k_a = k_b = aQ = bP = aGb$.

A possible attacker only has access to P and Q (and possibly G) which is insufficient to feasibly calculate k. However, the key exchange is vulnerable to man-in-middle attacks as P and Q are exchanged without authentication.

B. DHB-KEY Exchange Mechanism

DHB-Key is a modification of the basic ECDH key exchange mechanism using a static private number on one side and a shared ephemeral private number on the other side. The first half (Phase 1) of the ECDH key exchange is undertaken before deployment and establishes static keypairs for all sensor nodes. The second half (Phase 2) of the ECDH key exchange is executed periodically using an ephemeral keypair on the sink to establish new shared secrets on all nodes using a single broadcast message.

1) Phase 1: The sensor network consists of N sensor nodes and a sink. N private numbers $a_n (\forall 0 \leq n < N)$ are generated and corresponding public points P_n are calculated. Each sensor node s_n is configured with its a_n and a table on the sink is populated containing all P_n (see Figure 1). Phase 1 is carried out once only in a secure environment before network deployment:

1) ECDH parameters with curve base G are agreed.
2) All a_n and $P_n = Ga_n$ are calculated by the sink.
3) Each a_n is stored on the corresponding node s_n.
4) All P_n are tabled on the sink.

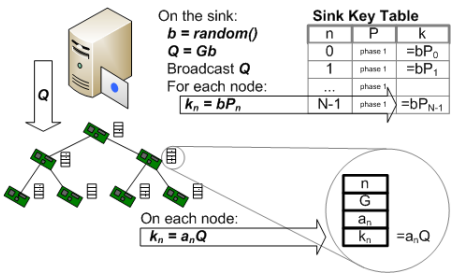

Figure 2. DHB-Key phase 2.

Because this is conducted before deployment, it is resistant to man-in-middle attacks. An attacker cannot obtain or modify any P_n and so will find it infeasible to imitate either party later on.

2) Phase 2: The sink generates a new ephemeral private number b and corresponding public point Q. The public point Q is distributed in the network using a single broadcast message (see Figure 2). All nodes s_n use this value Q to calculate a new individual shared secret k_n using their locally stored a_n. This process can be repeated periodically to set keys on all nodes.

1) The sink creates b and $Q = Gb$.
2) The public point Q is broadcast to all nodes.
3) Each node s_n recalculates the secret $k_n = a_nQ$.
4) The sink recalculates all secrets $k_n = bP_n$.
5) The secrets are shared as $k_n = a_nQ = bP_n = a_nGb_n$.

C. DHB-KEY Security

DHB-KEY is different to basic ECDH in two important aspects: (1) the sink's public point Q, derived from its ephemeral private number b, is common to all nodes in their production of shared secrets k_n and (2) the nodes private numbers a_n are not replaced in each key negotiation round. It has to be analysed if these differences represent a security risk.

Difference (1): Diffie-Hellman is used in other protocols in a one-to-many communication relation. For example, the Transport Layer Security (TLS) [6] protocol commonly used to secure the communication between web servers and Internet browsers can use DH key exchange. A web server can include a public point Q in a static certificate; all clients connecting to this server use this public point and their private number to create the shared secret $k_n = a_nQ$. TLS is widely used and considered to be safe. Thus, we conclude that this aspect of DHB-KEY does not represent a security risk.

Difference (2): The usage of static private numbers a_n on the nodes is similar to the Ephemeral-Static Mode described in RFC2631 [7]. This RFC defining the usage of DH in Internet protocols explicitly specifies a mode of operation in which one DH side uses a static private number and the other side uses a fresh private number for each negotiation. TLS specifies as well this mode of operation for key negotiation. Thus, we conclude that this aspect of DHB-KEY does not represent a security risk.

Other Security Aspects: An attacker might be able to compromise a node and obtain the stored private number a_n. From that point on the attacker is able to impersonate the compromised node and retrieve previous used keys. The problem of compromised nodes is not specific to DHB-KEY. Communication channels cannot be protected if endpoints are already compromised. Compromise of a_n does not damage the trust of the whole network, as that node remains unable to modify messages it forwards.

The symmetric cipher scheme used after the ECDH key exchange should offer equivalent or greater strength than ECDH. Otherwise, an attacker could recover the k_n and possibly a_n by breaking the used symmetric cipher instead of the DHB-KEY mechanism.

DHB-KEY does not authenticate the broadcasted public point Q in phase 2 and this can be exploited by an attacker for denial-of-service attacks. In particular, an attacker can maliciously inject false public points Q' which forces nodes to calculate invalid keys. Subsequent messages are dropped by the sink as wrong keys are used. Nevertheless, resources in the network are blocked by the attacker. For application scenarios as described in Section III this is not a problem. For other scenarios an authenticated broadcast message might be necessary. It is observed that existing broadcast authentication methods (such as μTESLA and DSA) are also subject to denial-of-service attacks[17].

V. Implementation

This section describes a combined routing and key distribution protocol named SecureTDRoute that uses the DHB-KEY mechanism and is tailored to the application scenario described in Section III. We decided to couple routing functionality and the DHB-KEY distribution mechanism as this allows us to implement a very efficient recovery strategy for lost key broadcasts (see Section VI). An implementation of SecureTDRoute for the TinyOS sensor node operating system is discussed as well in this section.

A. SecureTDRoute Routing Functionality

Sensor nodes in the network are organised in a tree structure which is rooted at the sink. Sensor nodes are statically deployed and each node is aware of its parent node in the tree structure and its child nodes. Two types of messages can be routed: (1) broadcasts from the sink and (2) unicast messages to the sink. There are two types of broadcast, *BCKEY* for key establishment and *BCMSG*

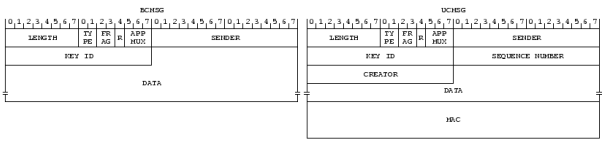

Figure 3. SecureTDRoute Packet Format

for application messages. There is one type of unicast message, *UCMSG*. The packet structure for *BCMSG* and *UCMSG* are shown in Figure 3. (*BCKEY* format is the same as *BCMSG*.) The packet formats are intended to be carried within a link-layer packet, such as TinyOS's ActiveMessage packets.

The $1byte$ length field indicates the length of the data field in bytes. The $2bit$ type field shows the data type contained in the packet (*BCKEY*, *BCMSG* or *UCMSG*). The $2bit$ frag field is used when data does not fit in one packet and fragmentation is necessary. The R bit is the recovery flag used to control broadcast failure recovery (explained later in detail). The $3bit$ appmux field is used to multiplex data between different applications running on the sensor node. The $2byte$ sender field shows the sender (last hop) of the packet. The $2byte$ creator field shows the original source of the packet and is also used to identify the correct key when the MAC is checked. The $2byte$ key ID field and the $4byte$ message authentication header are used for security purposes and are explained in the next paragraph.

Upon receiving a packet SecureTDRoute examines the sender field. If the packet is not sent by the parent or child nodes it is discarded. Thereafter the type field is examined. *BCKEY* and *BCMSG* messages are re-broadcasted and *UCMSG* messages are forwarded to the parent node. *BCKEY* data is used internally by SecureTDRoute to update key material. *BCMSG* data is passed to the application layer.

B. SecureTDRoute Security Functionality

Before deployment, the sink calculates the a_n and P_n for each node. The private number a_n is stored on each node and the public point P_n is tabulated on the sink. This constitutes phase 1. Periodically, the sink generates a new private number b and broadcasts the public point Q, with a unique key ID i as a *BCKEY* message. The sink calculates and stores the new shared secrets k_n and corresponding keys K_n. As each node receives the public point Q, the shared secret k_n is calculated at each node and the symmetric key K_n is derived (phase 2). Each node stores as well the current key ID i received in the *BCKEY* message.

Data sent by a node in a *UCMSG* message is secured by a $4byte$ Message Authentication Code (MAC). The MAC is computed over the all fields, except the MAC field itself, the sender and the R bit. The sequence number field is used to prevent replay attacks. The sink, on receipt of the message, inspects the key id field to select K_n and calculates the MAC. The two MACs are then compared and the message is dropped if they do not match. The sink might allow the usage of an old key as some nodes might not have received the latest *BCKEY* message for key updates.

C. Broadcast Failure Recovery

Packet losses are common in wireless sensor networks. For *UCMSG* messages an acknowledgment on the link layer can be used to detect a lost transmission and to initiate a re-transmission. For BCMSG broadcast messages this is not an option. A recovery mechanism is necessary to deal with lost broadcasts delivering important key material. SecureTDRoute applies the following solution: Each node inspects the key ID field when forwarding a unicast message upstream. If the key ID is lower than the locally stored key id it can be concluded that some node downstream has not received the latest key update. The node then creates a *BCKEY* message using locally stored public point Q and sends this message downstream. The recovery flag in the unicast message is set before the node forwards this message upstream to prevent nodes closer to the sink to initiate a repair broadcast as well. The sink might still decide to accept the unicast message with an old key if it is not deemed to be too old.

D. Implementation

SecureTDRoute was implemented for the MoteIV Tmote Sky node using TinyOS 2.0. The implementation fits into the existing TinyOS structure which allows us to reuse existing application code. SecureTDRouteC provides the standard TinyOS AMSend and Packet interfaces normally provided for applications in TinyOS. The key management is handled transparent to the application.

To implement ECDH on nodes, an existing implementation called EccM [2] was used. EccM uses the *sect163k1* curve parameters, resulting in a $21byte$ key size and a $42byte$ long public point Q. As a result, two messages to broadcast the public point Q are required as it was decided to comply with the TinyOS maximum payload size of $28byte$.

The critical bottleneck in ECDH is the time taken to perform scalar point multiplication. Longer calculation time consumes more energy and affects system responsiveness. An EccM key calculation time of $60s$ was measured on the Tmote Sky nodes. Recent work [10] shows that this calculation time can be reduced by 90%; however, this optimisation was not yet implemented in the prototype system. The calculation delay causes nodes to continue using old keys for a short period while the calculation completes. The sink therefore has to tolerate use of old keys for a period after a refresh.

During the key calculation, no other task can be carried out in TinyOS. This prevents communication and disables a mote's ability to report events and participate in the network. We use PLScheduler[9], a TinyOS 2.x extension for task pre-emption to counter this problem. The elliptic curve calculation is performed in a "low" task, which is automatically pre-empted by other tasks. Thus key calculation can be performed as a low priority background process.

E. SecureTDRoute Security

Our principle components are ECDH (*sect163k1* parameters using binary fields at $163bit$) for key exchange and AES ($128bit$, using CBC-MAC mode with the result truncated to $32bit$) for message authentication. ECDH and AES are 'unbroken' to our knowledge, and thus require an infeasible brute force attack. For ECDH, a parameter set which avoids weak curves is used. The CBC-MAC is secure as a fixed Interrupt Vector and fixed-length messages are used [13].

VI. Evaluation

The use of broadcast in DHB-KEY, rather than unicast, allows for several major benefits. (1) Key material can be feasibly cached within the network which reduces communication overhead penalties in failure scenarios. (2) Communication overhead is lower and better balanced, allowing the network to remain operational for longer. (3) The scheme aligns with WSN network properties. We make comparisons against the alternative of a symmetric key distribution scheme (herein referred to as the 'unicast scheme') where each node must be sent a separate message to establish a new key.

A. Key Distribution Message Overhead

Communication failures must be expected in wireless sensor networks and broadcast key update messages can be lost in transit. In the unicast scheme a different key update message is send to each node. In DHB-KEY a single message is broadcast to the whole network. One approach to reduce the impact of key update message losses is to retransmit cached messages within the network itself. Message caching quickly becomes infeasible in the unicast case as the necessary cache size is impractical in most WSN platforms. For example if a node is on the pathway to 100 nodes, it would need a 5KB cache if 50 byte messages were used. Nodes such as the Tmote Sky ship with 10KB of RAM, much of which is used by the application itself. In DHB-KEY the caching is feasible, because only a single message need be cached. This in-network caching therefore allows in-network recovery.

We implemented the physical intrusion detection application, in an office building, described in Section III. For the first experiment (Experiment A) the DHB-KEY protocol as specified in Section IV-B, including the described broadcast failure recovery mechanism, was used. For the second experiment (Experiment B), the alternative unicast key distribution is used using binary tree routing to allow unicast. In Experiment B, lost key distribution messages are re-transmitted from the sink. 14 nodes were implementing, each with door open/close contacts.

In case of perfect network conditions (no losses) 15 messages would be required in Experiment A to set keys on all nodes while in Experiment B 26 messages would be required (DHB-KEY requires 42% less messages). Clearly, DHB-KEY has a benefit in terms of required network resources. In the practical deployment where network links experience losses we recorded on average 16.8 ($\sigma = 2.22$) messages in Experiment A and 38.6 ($\sigma = 1.57$) messages in Experiment B (DHB-KEY requires 56% less messages in total). Thus, DHB-KEY is especially beneficial in realistic network settings where losses of key distribution messages must be compensated.

B. Network Lifetime

DHB-KEY can be compared with the unicast scheme by using the minimum number of messages that must be forwarded by a critical node at each key update. We use the term critical node to refer to the node closest to the sink in the situation where that node is the only node providing connectivity of the network to the sink. This node is critical because its failure will result in the disconnection of the entire network.

In DHB-KEY, this critical node is required to forward a minimum of one broadcast message and is required to compute its new key using DH which keeps the CPU active for a considerable time consuming the energy E_{DH}. In the unicast case, the critical node is required to forward a minimum of $N-1$ messages which costs $(N-1) \cdot E_{TX}$ energy but it does not have to spend significant time to set its key. Depending on the exact energy values E_{DH} and E_{TX} and the number of nodes N in the network either DHB-KEY or the unicast method will result in less energy cost on the critical node.

For our prototype implementation (see Section V) the following calculation applies. The MoteIV Tmote Sky with MSP430 MCU draws $1.9mA$ when busy and the CC2420 radio requires $19mA$ when active. Thus, each message costs $1.9mAs$ as we use the duty cycled FrameComm Medium Access Control Layer [14] which requires the radio to be active for $100ms$ to send a message (1% duty cycle, $41byte$ message size). A single ECDH key calculation with duration of $60s$ (see Section V) costs $114mAs$. The calculation is thus equivalent to 60 messages. In this example, the DHB-KEY method

is beneficial in networks with $N > 60$. Such a network will achieve a longer lifetime with DHB-KEY as the critical node consumes less energy. ECDH optimisations (for example [10]) will bring this number down further.

C. Network Properties

DHB-KEY is simple and requires less network features than the alternative unicast key distribution scheme. The network need only provide the capability to distribute a broadcast message from the sink to all sensor nodes. It is not necessary to maintain efficient unicast routes from the sink to each node at the time of key distribution. Many sensor networks experience strong fluctuation in link quality which can present difficulties in maintenence of valid optimal routes to all nodes [15]. The DHB-KEY key update messages can be distributed by broadcast and can thus be distributed in a network that has no fixed routing structure. For example, the DHB-KEY key update information can be piggy backed on a broadcast message used to setup a network structure for the following data transport from nodes to the sink.

Some sensor networks are optimised for asymmetric data flow, as most data flows towards the sink. In such networks it is common for very few resources to be provided to support downward data flow. Available network capacity is generally defined by the medium access control protocol. For example, DMAC [16] arranges the network such that messages are transported quickly towards the sink. For messages travelling in the opposite direction a small bandwidth is allocated and these messages incur a high latency. DHB-KEY is aligned with this asymmetric property of wireless sensor networks.

VII. Conclusion

The presented DHB-KEY scheme uses a single broadcast message to set individual keys on all nodes in the network. As shown, DHB-KEY has three main benefits. First, more network resources are available to application related tasks as they are not needed for key management. Second, energy consumption in the network is better balanced as communication is traded for computation. Third, the key exchange communication patterns match common sensor network properties which allows us to use DHB-KEY easily in practical deployments. We also show that DHB-KEY can support efficient broadcast failure recovery.

References

[1] A. Chung and U. Roedig. “Poster Abstract: DHB-KEY - A Diffie-Hellman Key Distribution Protocol for Wireless Sensor Networks”, Adjunct Proceedings of the 5th IEEE European Workshop on Wireless Sensor Networks (EWSN2008), Bologna, Italy, January 2008.

[2] D. Malan, M. Welsh and M. Smith. “A Public Key Infrastructure for Key Distribution in TinyOS Based on Elliptic Curve Cryptography”, Proceedings of IEEE Sensor and Ad Hoc Communications and Networks (SECON04), Santa Clara, California, October 2004.

[3] C.Karlof, N. Sastry and D.Wagner. “TinySec: A Link Layer Security Architecture for Wireless Sensor Networks”, Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, Baltimore, USA, November 2004.

[4] A. Wacker, M. Knoll, T. Heiber and K. Rothermel. “A New Approach for Establishing Pairwise Keys for Securing Wireless Sensor Networks”, Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems, San Diego, California, 2005.

[5] I. Chatzigiannakis, E. Konstantinou, V. Liagkou, P. Spirakis. “Design, Analysis and Performance Evaluation of Group Key Establishment in Wireless Sensor Networks”, Electronic Notes in Theoretical Computer Science (ENTCS), 2007.

[6] T. Dierks and C. Allen. “RFC 2246: The TLS Protocol Version 1.0”, 1999.

[7] E. Rescorla. “RCF 2631: The Diffie-Hellman Key Agreement Method”, 1999.

[8] A. Perrig, R. Szewczyk, V. Wen, D. Culler and J.D. Tygar. “SPINS: Security Protocols for Sensor Networks”, Proceedings of Mobile Computing and Networking, Rome, Italy, 2001.

[9] C. Duffy, U. Roedig, J. Herbert and C. Sreenan. “Adding Preemption to TinyOS”, Proceedings of the 4th workshop on Embedded networked sensors, Cork, Ireland, 2007.

[10] P. Szczechowiak, L. Oliveira, M. Scott, M. Collier and R. Dahab. “NanoECC: Testing the Limits of Elliptic Curve Cryptography in Sensor Networks”, European Conference on Wireless Sensor Networks (EWSN’08), Bologna, Italy, 2008.

[11] L. Eschenauer and V. Gligor. “A Key Management Scheme for Distributed Sensor Networks”, Proceedings of the 9th ACM conference on Computer and communication security, Washington DC, USA, 2002.

[12] Y. Zeng, J. Su, X. Yan, B. Zhao and Q. Huang. “LBKERS: A New Efficient Key Management Scheme for Wireless Sensor Networks”, The 3rd International Conference on Mobile Ad-hoc and Sensor Networks (MSN 2007), Bejing, China, 2007.

[13] M. Bellare, J. Kilian and P. Rogaway. “The Security of the Cipher Block Chaining Message Authentication Code”, Journal of Computer and System Sciences, 2000.

[14] J. Benson, T. O’Donnovan, U. Roedig and C. Sreenan. “Opportunistic Aggregation over Duty Cycled Communications in Wireless Sensor Networks”, Proceedings of the IPSN Track on Sensor Platform, Tools and Design Methods for Networked Embedded Systems (IPSN2008/SPOTS2008), St. Louis, USA, 2008.

[15] J. Benson, U. Roedig, T. O’Donovan and C. Sreenan. “Reliability Control for Aggregation in Wireless Sensor Networks”, Proceedings of the Second IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SENSEAPP2007), Dublin, Ireland, October 2007.

[16] G. Lu, B. Krishnamachari and C. Raghavendra. “An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Sensor Networks”, Proceedings of the International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks, Santa Fe, USA, April 2004.

[17] P. Ning, A. Liu and W. Du. “Mitigating DoS Attacks against broadcast Authentication in Wireless Sensor Networks”, ACM Transactions on Sensor Networks (TOSN), January 2008.