Poster Abstract: DHB-KEY - A Diffie-Hellman Key Distribution Protocol for Wireless Sensor Networks

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

***Abstract*—Many sensor network applications require secure communication between sensor nodes and the sink. This paper presents a key distribution scheme based on the well known Elliptic Curve Diffie-Hellman key exchange mechanism. The DHB-KEY scheme is performed in two stages. The first stage is carried out in a secure environment before network deployment. The second stage is carried out periodically using a single broadcast message. Each node arrives at a unique key it shares with the sink. This paper presents a first evaluation and a prototype implementation of the protocol. We have found that the presented key distribution approach uses energy and communication resources efficiently and has a low deployment complexity.**

I. Introduction and Motivation

Some sensor network applications require secure communication. In most 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, keys have to be negotiated between the sink and each node in the network.

A unique key should be used for each node to facilitate a simple key revocation mechanism in case that a node is deemed to have become untrustworthy. The use of symmetric keys is desirable as symmetric cryptographic algorithms require little computational effort on the resource constrained sensor nodes. To avoid cryptanalysis, it is desirable to refresh keys regularly. The key distribution should require as few messages as possible as i) network capacity should be available for the actual application task, and ii) communication is expensive in terms of energy.

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 (e.g. [7]). 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 [6]. Thus, a large portion of available resources in the network have to be spent on key distribution rather than application related tasks.

To overcome this, we have implemented DHB-KEY (first presented 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 us to minimize the communication overhead required to perform a key exchange.

II. Key Exchange Mechanism

Diffie-Hellman [2] is a well established method to agree a secret k between two parties, A and B, without transmitting the secret k over the insecure communication channel itself. The Elliptic Curve (EC) variation of DH provides similar security with significantly shorter keys. Thus, the key material that has to be exchanged is reduced, which is of importance in resource constrained wireless sensor networks.

Within ECDH [3], a key between A and B is established as follows. A and B agree a curve base G. A generates secret number a and the corresponding public point $P = Ga$. B generates secret number b and the public point $Q = Gb$. P and Q are exchanged over the insecure channel. A and B can now calculate the shared secret as $k = aQ = bP = aGb$. A possible attacker only has access to P and Q (and possibly G) which is not sufficient information to feasibly calculate k. However, an attacker might be able to intercept and modify all messages and negotiate a secret k_a with A and k_b with B. The standard DH is prone to such man-in-the middle attacks.

The basic idea of the DHB-KEY protocol is to use the previously described ECDH in the following way:

Phase 1 (Before Deployment)

1) All nodes are configured with the same EC parameters (including G).
2) a_n and $P_n = Ga_n$ are calculated for all n nodes.
3) a_n is stored on each corresponding node and all P_n are stored on the sink.

Phase 2 (After Deployment)

1) Regularly, the sink creates a new secret number b and public point $Q = Gb$.
2) The public point Q is then broadcast to all nodes.
3) Each node and the sink generate new keys $k_n = a_nQ = bP_n$.

III. Prototype Implementation

We implemented the previously outlined protocol as a TinyOS 2.x component which forms a layer between the application and network layer. Outgoing messages are protected with a message authentication code (MAC), which is added to the message within an additional header. The component accepts incoming key update messages and generates a new key as described in Section II. Our component provides AMSend and Packet interfaces and thus can be integrated easily in an existing application infrastructure.

Messages and Message Formats: The ECDH component adds security related header fields to a message. A one byte type field is used to distinguish control and data messages. When sending data, an n byte long MAC is computed and appended to a message (this is currently set to 4). When receiving, the sink's ECDH component verifies the MAC before passing the data to the application. Thus, the additional security features reduce the available payload by n+1 bytes.

The TinyOS default packet size forced us to use two broadcast messages to transmit the necessary 42 byte key material to nodes. A change of the default packet size would allow us to use a single broadcast message for key distribution. However, to avoid changes to the default TinyOS settings this possibility was not explored.

Cryptographic Functions: To implement ECDH on nodes, we have used parts of an existing elliptic curve application EccM [3], using its functionality to compute the shared secret. The sink is implemented in Java using the provided TinyOS Java interfaces and we use the BouncyCastle API [4] to implement elliptic curve cryptography functions on the sink.

Example Application Scenario: We evaluate the DHB-KEY protocol within a physical intrusion detection system. Nodes are equipped with sensors such as movement detectors and door/window contacts. Nodes forward messages through a tree structured network towards a sink. A very low network traffic will be observed most of the time. The sink has to be able to verify the authenticity and integrity of the message. An attacker is able to inject false messages into the network, but the sink can detect these and raise an intrusion alarm.

In this setting we use a sink initiated broadcast message once a day to refresh keys used on the nodes. Given the low traffic volume, the key refresh rate is deemed to be sufficient to prevent cryptanalysis.

IV. Analysis and Evaluation

Comparative evaluation: The proposed DHB-KEY method can be compared to a simple and straight forward mechanism (DIRECT method) in which individual key material is sent directly from the sink to each node.

Assume a tree network structure with N nodes. A node n_1 directly beneath the sink has to forward messages for $N-1$ nodes in a worst-case scenario. We now compare the situation of node $n_1$with each method, based on our motes: the MoteIV Tmote Sky with MSP430 MCU (1.9mA measured when busy) and CC2420 radio (~19 mA measured when active).

If a radio duty cycle method is used to preserve power, receiving messages requires no additional power, but transmission has to be extended to avoid the need to synchronise clocks. With a duty cycle of 1%, an epoch length of $100ms$ and a message size of 41 bytes, the radio needs to send for an average of a full epoch ($100ms$) to forward each message.

Using the DIRECT method, the node n_1 has to forward $N-1$ key update messages from the sink. Thus, it will be active for $(N-1)\cdot 100ms$ to forward the key update material.

Using the DHB-KEY method, the node n_1 needs to forward only one broadcast message (respectively two messages in our prototype implementation). However, after the broadcast message arrives, n_1 must perform the DH calculation to obtain the secret. This calculation takes about $60s$ in our prototype.

Communication effort can be traded for calculation effort. The fixed cost of the calculation means that there is an N where the calculation cost is less than the cost of forwarding $N-1$ messages. The exact point where DHB-KEY becomes beneficial depends on network implementation specifics such as the medium access control protocol and the duty cycle.

Each message costs 1.9 mAs (milliamp/second) to send. A single key calculation costs 114 mAs. The calculation is thus equivallent 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 as n_1 will ensure network connectivity for a longer time period.

DHB-KEY has additional benefits. The distribution of the broadcasts can be faster than distributing key material to all nodes individually. Thus, a key change can avoid clogging the network, keeping it available for application-related network traffic. A sensor network is also often optimized to transport bulk data from the nodes towards the sink, not to transport large amounts of data from the sink to the nodes. DHB-KEY thus fits the communication patterns of a sink-tree sensor network better.

V. Conclusion and Future Work

The proposed DHB-KEY scheme has the benefit that unique keys can be set for all nodes in the network by sending just a broadcast message. The scheme can offer a number of other benefits such as bandwidth availability, management and energy consumption. Our future work involves completion of the implementation to record more performance data, enhancing the performance of the key calculation, recovery mechanisms for broadcast failures and further security analysis.

References

[1] Poster: Efficient Key Establishment for Wireless Sensor Networks Using Elliptic Curve Diffie-Hellman. Tony Chung and Utz Roedig. 2007.

[2] New Directions in Cryptography. Whitfield Diffie, Martin E. Hellman. 1976.

[3] A Public Key Infrastructure for Key Distribution in TinyOS Based on Elliptic Curve Cryptography. Malan, Welsh, Smith.

[4] BouncyCastle API. http://www.bouncycastle.org/

[5] TinySec: A Link Layer Security Architecture for Wireless Sensor Networks. Karlof, Sastry, Wagner. 2004.

[6] A New Approach for Establishing Pairwise Keys for Securing Wireless Sensor Networks. Wacker, Knoll, Heiber and Rothermel. 2005.

[7] Design, Analysis and Performance Evaluation of Group Key Establishment in Wireless Sensor Networks. Chatzigiannakis, Konstantinou, Liagkou, Spirakis. 2007.