The Energy Cost of Cryptographic Key Establishment in Wireless Sensor Networks (Extended Abstract)[*]

Johann Großschädl
jgrosz@iaik.tugraz.at

Alexander Szekely
aszekely@iaik.tugraz.at

Stefan Tillich
stillich@iaik.tugraz.at

Graz University of Technology
Institute for Applied Information Processing and Communications
Inffeldgasse 16a, A–8010 Graz, Austria

ABSTRACT

Wireless sensor nodes generally face serious limitations in terms of computational power, energy supply, and network bandwidth. Therefore, the implementation of effective and secure techniques for setting up a shared secret key between sensor nodes is a challenging task. In this paper we analyze and compare the energy cost of two different protocols for authenticated key establishment. The first protocol employs a lightweight variant of the Kerberos key transport mechanism with 128-bit AES encryption. The second protocol is based on ECMQV, an authenticated version of the elliptic curve Diffie-Hellman key exchange, and uses a 256-bit prime field GF(p) as underlying algebraic structure. We evaluate the energy cost of both protocols on a Rockwell WINS node equipped with a 133 MHz StrongARM processor and a 100 kbit/s radio module. The evaluation considers both the processor's energy consumption for calculating cryptographic primitives and the energy cost of radio communication for different transmit power levels. Our simulation results show that the ECMQV key exchange consumes up to twice as much energy as Kerberos-like key transport.

Keywords: Wireless networking, security protocols, key establishment, cryptography, energy evaluation.

1. INTRODUCTION

Wireless sensor networks often operate in hostile environments or collect sensitive data (e.g. in health care), which calls for effective measures to protect them against security threats like unauthorized access, manipulation of data, and denial of service [1]. However, wireless sensor networks are an extremely demanding environment for security architects due to the resource-constrained nature of battery-operated sensor nodes (limited energy supply, modest computational power, small memory/storage, and low-bandwidth network connectivity). In the recent past, a number of security architectures and protocols for wireless sensor networks have been proposed, taking the special characteristics of sensor nodes into account. Authentication and key establishment are the two most essential security services to maintain the proper operation of a sensor network.

A simple approach for key establishment is to load one or more keys onto each sensor node prior to deployment. Most practical security protocols based on pre-deployed keying use either a single network-wide key shared by all sensor nodes or a set of keys randomly chosen from a key pool so that two nodes will share (at least) one key with a certain probability [3]. These protocols are easy to implement and entail only little overhead since no complex computations have to be performed. However, most security architectures based on pre-deployed keys suffer from problems caused by poor scalability and vulnerability to node capture. A second family of key establishment protocols uses a trusted third party (e.g. the base station) to set up a secret key between two sensor nodes. Each node shares a long-term secret key with the base station, similar to the well-known Kerberos protocol [7]. Examples of "Kerberos-like" security protocols for sensor networks include the node-to-node key establishment schemes based on SNEP [9] and PIKE [2]. All these protocols have in common that they are built on symmetric cryptography and entail high communication cost.

Key establishment in sensor networks can also be realized with protocols which use public-key cryptography to set up a shared secret key between two nodes. The most important key exchange protocol was proposed by Diffie and Hellman in 1976 and is usually implemented using the multiplicative group of a finite field of prime order. Alternatively, it is also possible to embed the Diffie-Hellman key exchange into an additive group like the group of points on an elliptic curve defined over a finite field [5]. Typical group sizes range from 1024 to 2048 bits for $\mathbb{Z}_p^*$ and from 160 to 256 bits when an elliptic curve group is used. These large group orders make Diffie-Hellman key exchange highly computation-intensive and, hence, energy consuming. Carman et al. reported in [1] that the energy cost of a 1024-bit modular exponentiation—such as needed for Diffie-Hellman key exchange—is about 14.6 mJ on a Rockwell WINS node equipped with a 133 MHz StrongARM microprocessor. For comparison, the encryption of a 128-bit data block using a symmetric cipher like the AES requires only 2.17 μJ. Hodjat and Verbauwhede [6] analyzed and compared the energy cost of elliptic curve

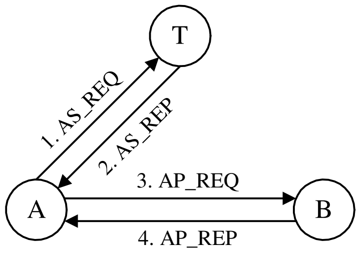
[*]The research described in this paper has been supported by the Austrian Science Fund under grant P16952-NO4 and by the European Commission under grant FP6-IST-033563 (project SMEPP) and IST-2002-507932 (ECRYPT).

Diffie-Hellman (ECDH) key exchange and Kerberos-like key establishment on a Rockwell WINS node. They considered in their study both the energy required for calculating cryptographic primitives and the energy cost of communication between the sensor nodes. It was concluded in [6] that, for comparable security levels, ECDH key exchange consumes 67 times more energy than Kerberos key transport.

The heavy energy requirements of public-key algorithms reported in [1, 6] have raised serious concerns about their feasibility in wireless sensor networks. However, it must be considered that the experiments documented in [1] and [6] were conducted in 2000 and 2002, respectively, and since then, enormous progress has been made in the efficient implementation of public-key cryptography, especially elliptic curve cryptography [5]. This progress makes it necessary to completely re-evaluate the energy requirements of elliptic curve cryptosystems. In the present paper we demonstrate that the energy cost of elliptic curve cryptography—when implemented with state-of-the-art algorithms—is far lower than reported in previous work. Our experimental results clearly show that key exchange protocols using elliptic curve systems are feasible for wireless sensor networks.

2. KERBEROS KEY TRANSPORT

Kerberos is a network authentication system that uses a *trusted third party* to authenticate two entities (i.e. to prove their identity to one another) by issuing a shared session key between them [7]. The session key is actually generated by the trusted third party (in the following abbreviated by T) and then securely delivered to the two entities wishing to establish a shared secret. Each entity on the network shares a unique long-term secret key with T, which enables the entities to verify that messages from T are authentic. Similarly, knowledge of the long-term key also serves to prove an entity's identity to T. In a sensor network, the unique long-term key that each node shares with T is generally pre-deployed [1]. The trusted third party can either be a single centralized entity like the base station or a cluster head [1], or distributed among trusted sensor nodes [2].



1. AS_REQ: A, B, n_A
2. AS_REP: $\{ k_{AB}, B, t_S, t_E, n_A \} k_{AT}$, $\{ k_{AB}, A, t_S, t_E \} k_{BT}$
3. AP_REQ: $\{ k_{AB}, A, t_S, t_E \} k_{BT}$, $\{ A, t_A \} k_{AB}$
4. AP_REP: $\{ t_A \} k_{AB}$

Figure 1: Simplified Kerberos protocol [4].

Figure 1 illustrates the message transfers between entities A, B, and T, assuming that A wishes to establish a session key with entity B. An expression of the form $\{X\}k$ means that message X is encrypted using the key k. Kerberos, as specified in [7], is a full-featured protocol whose messages can have a size of several kilobytes, which poses a problem for sensor networks since radio communication drains the nodes' battery. Therefore, we conduct our energy evaluation with a "lightweight" version of the Kerberos protocol with short messages as summarized below (see [4] for details).

In the first message (AS_REQ) entity A asks the trusted third party T for a session key that enables A to securely communicate with B. Thereupon, T generates a session key k_{AB} and assembles a reply message (AS_REP) consisting of a *ticket* and other data as illustrated in Figure 1. The ticket contains the freshly generated session key k_{AB} and is encrypted in the long-term key k_{BT} shared between B and T. Besides the ticket, the AS_REP message also includes the session key k_{AB} in a form readable by A, i.e. encrypted in the long-term key k_{AT} shared between A and T. Having received AS_REP, entity A decrypts the non-ticket portion of the message to obtain the session key k_{AB}. Next, A produces an *authenticator*, encrypts it using the key k_{AB}, and sends it along with the ticket to entity B (message AP_REQ in Figure 1). Entity B first decrypts the ticket using its long-term key k_{BT} and extracts the session key k_{AB}. The session key enables B to decrypt the authenticator from the AP_REQ message, which, if successful, proves A's identity since only a legitimate entity possessing k_{AB} can generate a valid authenticator. Finally, message AP_REP, sent from B to A, confirms B's true identity and completes the mutual authentication.

3. ECDH / ECMQV KEY EXCHANGE

The elliptic curve Diffie-Hellman (ECDH) key exchange is the elliptic curve analogue of the classical Diffie-Hellman scheme in the group $\mathbb{Z}_p^*$ [5]. Similar to other elliptic curve cryptosystems, ECDH operates on an elliptic curve defined over a finite field. The ECDH protocol requires the involved entities to compute a so-called *point multiplication*, i.e. an operation of the form $k \cdot P$ where k is an integer and P is a point on the curve [5]. A point multiplication is performed through arithmetic operations in the underlying finite field (in particular field multiplication), which makes ECDH very computation-intensive since the fields used in elliptic curve cryptography have an order of at least 160 bits.

Before an ECDH key exchange can be accomplished, the two involved entities have to agree upon a set of domain parameters. These parameters include the elliptic curve to be used, the underlying finite field, a generator G of the group, and the group order n. Let A and B be two entities wishing to establish a shared secret. First, entity A chooses a random secret number a with $2 \leq a \leq n-2$, calculates $U = a \cdot G$, and sends U to entity B. Entity B also chooses a random secret number b in the range $[\,2, n-2\,]$, calculates $V = b \cdot G$, and sends V to entity A. Entity A is now able to compute the shared secret key as $K = a \cdot V = a \cdot b \cdot G$ and entity B can compute $K = b \cdot U = b \cdot a \cdot G$. Both entities have agreed on the same key because $a \cdot b \cdot G = b \cdot a \cdot G$. In summary, an ECDH key exchange requires to perform four point multiplications and to send two messages.

The standard ECDH protocol, as described before, does not authenticate the two entities A and B, and is therefore vulnerable to man-in-the-middle attacks. An example of a key exchange protocol providing *implicit authentication* is the Menezes-Qu-Vanstone (MQV) protocol, and its elliptic curve analogue, the ECMQV protocol [8]. The two entities participating in the ECMQV key exchange are assumed to have both a *static* (i.e. long-term) public/private key pair and an *ephemeral* (i.e. short-term) key pair. A key pair in ECMQV consists of a private key, which is just a random number x, and a public key, given by the product of the random number x and the base point G. The static key pair is valid for a certain period of time, whereas new ephemeral keys are generated for each run of the protocol. A shared secret is derived using the static keys and the ephemeral keys, which guarantees that each protocol run between two entities A and B produces a different shared secret. Before the actual ECMQV key agreement can take place, the two involved entities have to exchange the public part of their static keys. Then, each entity generates an ephemeral key pair (which requires to compute a point multiplication) and sends the public part of this key to the other entity, similar to the standard ECDH protocol. Now, in order to derive a shared secret key, each entity has to calculate a so-called *multiple point multiplication* of the form $k \cdot P + l \cdot Q$, which is only slightly more costly than calculating $k \cdot P$ if Shamir's trick is used [5]. We refer to [4, 5] for a detailed description of ECMQV key exchange.

In summary, every run of the ECMQV protocol requires to calculate two point multiplications of the form $k \cdot P$, two multiple point multiplications of the form $k \cdot P + l \cdot Q$, and to send two messages, provided that the public parts of the static keys have already been exchanged.

4. RESULTS AND CONCLUSIONS

The overall energy cost of key establishment in a wireless sensor network is determined by the energy required for the execution of cryptographic primitives and the energy cost of radio communication between the involved entities. We conducted the energy evaluation of our lightweight Kerberos protocol and the ECMQV protocol on a Wireless Integrated Network Sensor (WINS) node from Rockwell Scientific. The WINS node is equipped with a StrongARM microprocessor clocked at 133 MHz and features a 100 kbit/s radio module [1]. According to [10], the WINS node has an average power consumption of 360 mW when the processor is active. The transmission of a single bit of data requires, depending on the transmit power level, an energy of between 7.71 μJ and 10.8 μJ on the sending node, and 7.52 μJ on the receiving node [10]. Thus, the overall energy required for sending and receiving of data ranges from 15.2 μJ/bit to 18.3 μJ/bit.

Protocol	Comp.	Msg. transfer	Total energy
Kerberos	0.1 mJ	39.5–47.5 mJ	39.6–47.6 mJ
ECMQV	51.8 mJ	27.2–32.8 mJ	79.0–84.6 mJ

Table 1: Energy analysis of Kerberos and ECMQV.

The lightweight Kerberos protocol described in Section 2 requires to send four messages between entities A, B, and T. According to the analysis in [4], the payload of the four messages is 1568 bits altogether. In addition to the actual payload, each message contains other data like a protocol ID, a message ID, a checksum, as well as low-level headers and footers, which increase the size of each message by 256 bits. Thus, a total of 2592 bits has to be transmitted, and consequently the communication energy cost of Kerberos is between 39.5 mJ and 47.5 mJ (the exact value depends on the transmit power level). As illustrated in Figure 1, three of the four Kerberos messages are encrypted. When using 128-bit AES, the overall energy required for encryption and decryption of the messages is less than 0.1 mJ (see [4] for a detailed analysis). In summary, the Kerberos protocol is characterized by high communication energy cost, while the energy needed for en/decryption is almost negligible.

We implemented the ECDH/ECMQV key exchange using a 256-bit prime field as underlying algebraic structure in order to match the security level of Kerberos key transport with 128-bit AES encryption. A point multiplication over a 256-bit prime field takes approximately $4.25 \cdot 10^6$ clock cycles on a StrongARM processor when implemented as described in [4]. The execution time of a multiple point multiplication of the form $k \cdot P + l \cdot Q$ is about $5.31 \cdot 10^6$ cycles [4]. Given a clock frequency of 133 MHz and a power consumption of 360 mW, the corresponding energy values are 11.5 and 14.4 mJ, respectively. Every run of the ECMQV protocol requires to perform two point multiplications $k \cdot P$ and two multiple point multiplications $k \cdot P + l \cdot Q$, resulting in an overall computation energy cost of 51.8 mJ. Moreover, two messages have to be exchanged, each with a payload of 640 bits (see [4] for further details). Adding an overhead of 256 bits to each message, the total communication energy cost of ECMQV is between 27.2 mJ and 32.8 mJ.

Putting it all together, the overall energy cost of Kerberos key establishment is between 39.6 mJ and 47.6 mJ, while ECMQV key exchange requires an energy of between 79.0 mJ and 84.6 mJ (see Table 1). In other words, the energy consumption of ECMQV and Kerberos differs merely by a factor of between 1.78 (for high transmit power) and 1.99 (for low transmit power). Thus, key establishment protocols using public-key cryptography can no longer be considered prohibitively expensive in terms of energy consumption.

5. REFERENCES

[1] D. W. Carman, P. S. Kruus, and B. J. Matt. Constraints and Approaches for Distributed Sensor Network Security. Technical Report #00-010, Network Associates Inc., 2000.

[2] H. Chan and A. Perrig. PIKE: Peer intermediaries for key establishment in sensor networks. In *Proceedings of the 24th IEEE Conference on Computer Communications (INFOCOM 2005)*, vol. 1, pp. 524–535. IEEE, 2005.

[3] L. Eschenauer and V. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, pp. 41–47. ACM Press, 2002.

[4] J. Großschädl, A. Szekely, and S. Tillich. The energy cost of cryptographic key establishment in wireless sensor networks. Cryptology ePrint Archive, Report 2007/003. Available for download at `http://eprint.iacr.org`, 2007.

[5] D. Hankerson, A. J. Menezes, and S. A. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer Verlag, 2004.

[6] A. Hodjat and I. Verbauwhede. The energy cost of secrets in ad-hoc networks. In *Proceedings of the 5th Workshop on Wireless Communications and Networking*. IEEE, 2002.

[7] J. T. Kohl and B. C. Neuman. The Kerberos Network Authentication Service (Version 5). Internet Engineering Task Force (IETF), Internet Draft RFC 1510, Sept. 1993.

[8] L. E. Law et al. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, 28(2):119–134, Mar. 2003.

[9] A. Perrig et al. SPINS: Security protocols for sensor networks. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MOBICOM 2001)*, pp. 189–199. ACM Press, 2001.

[10] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava. Energy-aware wireless microsensor networks. *IEEE Signal Processing Magazine*, 19(2):40–50, Mar. 2002.