Fast Authenticated Key Establishment Protocols for Self-Organizing Sensor Networks

Qiang Huang[1], Johnas Cukier[2], Hisashi Kobayashi[1], Bede Liu[1] and Jinyun Zhang[2]
1. Department of Electrical Engineering, Princeton University, Princeton, NJ 08544, USA
2. Mitsubishi Electric Research Laboratories, 201 Broadway, Cambridge, MA 02139, USA

ABSTRACT
In this paper, we consider efficient authenticated key establishment protocols between a sensor and a security manager in a self-organizing sensor network. We propose a hybrid authenticated key establishment scheme, which exploits the difference in capabilities between security managers and sensors, and put the cryptographic burden where the resources are less constrained. The hybrid scheme reduces the high cost public-key operations at the sensor side and replaces them with efficient symmetric-key based operations. Meanwhile, the scheme authenticates the two identities based on public-key certificates to avoid the typical key management problem in pure symmetric-key based protocols and maintain a good amount of scalability. The proposed scheme can be efficiently implemented on Mitsubishi's M16C microprocessor in 5.2Kbyte code/data size, and achieve a total processing time of 760 ms on sensor side, which is better than all the other public-key based key establishment protocols we have studied. We also present its modified version with a faster speed but more communication overhead.

Categories and Subject Descriptors
C.2.0 [**Computer-Communications Networks**]: Security and Protection; C.2.1 [**Network Architecture and Design**]: Wireless Communication

General Terms
Design, Security, Performance

Keywords
Key Establishment, Sensor Network, Security, Elliptic Curve Cryptography

1. INTRODUCTION
Self-organizing sensor networks have been proposed to support dynamic scenarios and facilitate large-scale, real-time data processing in complex environments. Self-organizing sensor networks can be quickly and inexpensively set up as needed since they do not require any centralized administration or fixed infrastructure like a base station or access points. The IEEE 802.15.4 Low-Rate Wireless Personal Area Network Standard [13] specifies the physical layer and medium access control layer of a low data rate, ultra low power and low cost sensor network. Target applications include natural disaster control, health care, battlefield service, oil site operation, rescue missions, etc. In these and other vital or security-sensitive deployments, secure and fast transmission of sensitive digital information over the sensor network is essential. The use of encryption or authentication primitives between two sensor devices require an initial link key establishment process, which must satisfy the low power and low complexity requirement. The very ad hoc nature of sensor networks and the cost constraints that are often imposed on them make these networks difficult to secure. Communications cannot rely on the online availability of a fixed infrastructure or central administrator, thus decentralized online key management becomes a necessity.

The IEEE 802.15.4 standard defines two physical device types, a Full-Functional Device (FFD) and a Reduced-Functional Device (DRF). An RFD takes on the logical role of an end device, e.g. a sensor, while an FFD can also take the role of coordinator, router or security manager. A security manager is an FFD granted special capabilities to assist in provisioning link keys to other devices. A security manager may be portable, so it is used to configure sensors on-site. The security manager should first establish a link key with a sensor before it can install link keys into that sensor for secure communicating with other devices inside the cluster. There is also an off-site central authority (CA), which is kept physically secure and used to preload initial authentication data (certificates) to security managers and sensors offline.

This paper focuses on the initial link key establishment between a sensor node (RFD) and a security manager (FFD). RFDs will have less computational resources and memory capacity than FFDs. Implementing security solutions in an RFD poses the largest challenge, due to strict implementation constraints. One way to accomplish the initial link key establishment task is to pre-install a link key table into each device, and a sensor needs to keep a different link key for each different security manager. However, sensor networks may be highly versatile, involving temporary communications between devices that may have never met before. Thus we cannot predict and install all link keys for devices before they join the network, especially for large-scale sensor networks. An alternative way is to use a shared group key that is pre-loaded into each device [3]. However, a common group key poses a security risk if any one device is compromised. The use of asymmetric keys along with digital certificates to establish individual link keys can help reduce this risk. Public-key based techniques, along with digital certificates, restrict the impact of key compromise to the compromised node itself, rather than to all its key-sharing parties.

WSNA'03, September 19, 2003, San Diego, California, USA.

Public-key operations are quite expensive though, which remains a problem for portable devices with limited computation resources and power supplies. In recent years, symmetric-key based key agreement protocols have gained popularity due to the small computation overhead. However, the key management for pure symmetric-key based system is complicated, either a key distribution center (KDC) is online involved or a large number of symmetric keys need to be pre-loaded into devices. Both methods reduce the scalability of self-organizing sensor networks. Section 2 gives an introduction on prior key establishment protocols.

In section 3 of this paper, we propose a hybrid authenticated key establishment scheme, which is based on a combination of elliptic curve cryptography (ECC) and symmetric-key operations. The motivation is to exploit the difference in capabilities between security managers and sensors, and put the cryptographic burden where the resources are less constrained. Sensors are much more battery and computational resources limited. However, the security manager means powered and more computational powerful. The hybrid key establishment protocol reduces the high cost elliptic curve random point scalar multiplications at the sensor side and replaces them with low cost and efficient symmetric-key based operations. On the other hand, it authenticates the two identities based on elliptic curve implicit certificates [7] to avoid the typical key management problem in pure symmetric-key based protocols.

Section 4 provides the security analysis of the proposed hybrid key establishment protocol. Section 5 analyzes the computation complexity, communication complexity, storage requirements and code size for our protocol. In section 6, we present a modified version of our hybrid protocol, with even less computation cost on sensor side, but more communication overhead. In section 7, we compare our hybrid protocol and its modified version to other public-key based key establishment protocols in processing time and bandwidth requirements. The paper concludes in section 8. The appendices provide the detail analysis of the computation complexity of integer modular multiplication and elliptic curve point multiplication.

2. RELATED WORK

Various public-key based key establishment protocols are used to set up symmetric link keys, such as Shamir's three-pass protocol, U.S. Patent No. 4748668, the Diffie-Hellman public-key protocol, U.S. Patents No. 4,200,770, the Aziz-Diffie protocol [2], the Beller-Chang-Yacobi's protocol [4], etc. Boyd and Mathuria survey the previous work on key distribution and authentication for low power devices in wireless environments [6]. Hubaux et al. [11] consider an ad hoc network with nodes powerful enough for performing asymmetric cryptographic operations and propose a peer-to-peer authentication protocol based on public-key certificates. Zhou and Hass propose to secure ad hoc networks using asymmetric cryptography [30], focusing on distributing the role of the CA over some or all devices in the network. The main approach is based on threshold cryptography [25] and allowing specific coalitions of devices to act together as a source of trust to issue public-key certificates. Their approach decentralized the online key management, as necessary in mobile ad hoc networks. Unfortunately, all above schemes use asymmetric cryptography and hence, are expensive for the sensor network environments.

The computation complexity and power consumption of symmetric-key based protocols are negligible when compared with public-key operations. However, the key management for symmetric key based protocols is complicated, and is always subject to attack by adversaries. Basagni et al. propose to use a network-wide symmetric key to secure an ad hoc network [3]. While this group key approach is efficient, it does not protect against compromise of a single node. Perrig et al. present the security protocol SPINS [21], and a trusted third party is involved to assist node-to-node key agreement. However, how to establishment a master key between a sensor node and the trusted third party is left as an open problem. If a single trusted third party is used, e.g. a KDC, then it must be online involved as a central administrator, which reduces the scalability and self-organizing ability of the sensor networks. If multiple trusted devices are involved to configure sensors on site, they are functioning similarly as security managers in our system. However, we cannot pre-load all master keys needed for the sensor to communicate to all security managers, due to the high mobility and a limited memory space. Public-key based protocols give more flexibility and scalability especially in large sensor networks where new devices keep entering the cluster.

Recently several schemes have been proposed to offload certain public-key cryptographic computation to servers and have the low-end devices to do less work. Modadugu et al. propose to offload the heavy computation for finding an RSA key pair to untrusted servers [19]. In [28], Wong and Chan proposed two key exchange protocols to reduce computation on the mobile client side, the server-specific MAKEP and the linear MAKEP. In the server-specific MAKEP scheme, a server can impersonate a client once they run the key exchange protocol. In the linear MAKEP, the number of times that the client is able to run the protocol is determined by how many public-private key pairs it stores, and hence the protocol does not scale well. The same authors present a modified scheme with a better scalability [29], but the client has to perform modular exponentiation to generate the ephemeral public key $\beta=g^b \bmod p$, where g and p should be greater than 512 bits for security reason, and b is 160 bits. The processing time to generate an ephemeral key on Palm V is 36 sec for 512-bit modulus and 144 sec for 1024-bit modulus. A sensor node cannot afford to do such expensive computation. Also, the certificate of the server needs to be verified online, and the computation cost is not negligible. A solution may be to preload some ephemeral keys (e.g. in [28]) or offload the computation to an untrusted server. Then the scalability of the protocol is reduced. Jakobsson and Pointcheval [14] proposed to improve efficiency by using pre-computation. However, the protocol is shown to be susceptible to a variant of interleaving attacks in [29]. Furthermore, the pre-computation is not applicable in the self-organizing sensor networks we envision, where a sensor can in general not predict which party it deals with.

In recent years, ECC based key agreement protocols are designed to use in constrained mobile environments, due to the property of small key sizes, such as the ECMQV protocol with ECC X509 certificates [27] and implicit certificates [7], the ECDSA authenticated key exchange protocol [1], the Elliptic-Curve Diffie-Hellman Ephemeral (ECDHE) protocol [24], etc. However, the computation load of all the above ECC based protocols consists of at least two expensive elliptic curve random point scalar multiplications on each participating entity. In this

paper, we focus on reducing the elliptic curve random point scalar multiplications on sensor nodes by offloading the computation burden to more powerful security managers, and replacing the expensive public-key operations by efficient symmetric-key operations. We also use the elliptic curve implicit certificate to avoid the typical key management problem in pure symmetric-key based protocols.

3. A HYBRID AUTHENTICATED KEY ESTABLISHMENT PROTOCOL

ECC is used in our protocol to perform security functions on sensors with limited computing resources. Compared with other public key crypto algorithms, much smaller key lengths are required with ECC to provide a desired level of security, which means faster processing speed, smaller communication complexity, in addition to smaller key storage requirements.

To prevent the impersonation attack, we use certificates in our key-establishment protocol, which provide a mechanism to check cryptographically to whom the public key belongs and if the device is a legitimate member of a particular network. A certificate is simply a public key together with the device ID and certification expiration date, signed by CA. The use of a trusted interface to pre-establish a certificate and root key in a device thwarts both active and passive attacks in subsequent key establishment protocols. The certificates are acquired before each device joins the network through an out-of-band interface. We use the elliptic curve implicit certificate scheme [27], because of the resulting low communication complexity, which is a dominant factor for low bit transmission channels in sensor networks.

First, an elliptic curve E defined over $GF(p)$ (where p is the characteristic of the base field) with suitable coefficients and a base point P of large order n is selected and made public to all users. CA selects a random integer q_{CA} as its static private key, and computes the static public key $Q_{CA} = q_{CA} \times P$. To obtain a certificate and the static private-public key pair, the sensor U randomly selects a temporary key pair (g_U, G_U) and sends G_U to CA via a secure out-of-band interface. CA verifies U's identity and the authenticity of the request received from U. CA also selects a temporary key pair (g_{CA}, G_{CA}) and computes the elliptic curve point $B_U = G_U + G_{CA}$. The implicit certificate IC_U for U is constructed as the concatenation of CA's static public key Q_{CA}, the device identity ID_U, the elliptic curve point B_U and the certification expiration date t_U, i.e., $IC_U = (Q_{CA}, ID_U, B_U, t_U)$. CA then applies a one-way hash function H on IC_U and derives an integer $e_U \in [2, n-2]$ from $H(IC_U)$ following the conversion routine described in Section 4.1.3 of [7]. Finally, CA computes U's private-key reconstruction data $s_U = g_{CA} e_U + q_{CA} \pmod n$, U's public key $Q_U = e_U B_U + Q_{CA}$, and sends s_U and IC_U back to U. After U receives the implicit certificate from CA, it computes the hash value $H(IC_U)$ and derives an integer e_U from $H(IC_U)$ following the conversion routine described in Section 4.1.3 of [7]. U also computes its static private key $q_U = s_U + g_U \cdot e_U \pmod n$ and its public key $Q_U = q_U \times P$. U then reconstructs the public key $\hat{Q}_U = e_U B_U + Q_{CA}$. If $\hat{Q}_U = Q_U$, U accepts the certificate and outputs the static key pair (q_U, Q_U); otherwise it rejects the certificate. By repeating the very same process, the security manager V acquires its certificate IC_V and static key pair (q_V, Q_V).

U		CA (q_{CA}, Q_{CA})
$g_U \in [2, n-2]$		$g_{CA} \in [2, n-2]$
$G_U = g_U \times P$		$G_{CA} = g_{CA} \times P$
Send	$G_U, ID_U \rightarrow$	Receive
		$B_U = G_U + G_{CA}$
		$IC_U = (Q_{CA}, ID_U, B_U, t_U)$
		$H(IC_U) \rightarrow e_U \in [2, n-2]$
		$s_U = g_{CA} e_U + q_{CA} \pmod n$
		$Q_U = e_U B_U + Q_{CA}$
Receive	$\leftarrow (s_U, IC_U)$	Send
$H(IC_U) \rightarrow e_U$		
$q_U = s_U + g_U e_U \pmod n$		
$Q_U = q_U \times P$		
$\hat{Q}_U = e_U B_U + Q_{CA}$		
$Q_U = \hat{Q}_U$?		

Figure 1. Implicit certificate generation process.

The certificate generation processes for sensor U and security manager V are performed offline as shown in Figure 1, before they join the network, and may be assisted by a trusted computation server. When they first communicate to each other, they execute our hybrid key establishment protocol as shown in Figure 2:

1. U and V send to each other their implicit certificates. The content of the certificate is verified at the other side, including the device identity and the validity period. If any check fails, the protocol is terminated.

2. V computes the hash value $H(IC_U)$ and derives an integer e_U from $H(IC_U)$ following the conversion routine described in Section 4.1.3 of [7]. V then obtains U's public key $Q_U = e_U B_U + Q_{CA}$. After performing the certificate processing operation, V can conclude that Q_U is genuine, provided that U later evidences knowledge of the corresponding private key q_U.

3. U selects a k-bit random number c_U as its link key contribution and a random $160-k$ bit integer r. U calculates its ephemeral private key $d_U = H(c_U \| r)$ and ephemeral public key $D_U = d_U \times P$, where H is a cryptographic hash function to map a binary string to a random integer $\in [2, n-2]$. U verifies V's certificate and obtains V's public key the same way as V does, but instead of computing Q_V directly, U computes $R = d_U \times Q_V = (d_U e) \times B_V + d_U \times Q_{CA}$. U can conclude that R is

Sensor U		Security manager V
(q_U, Q_U)		(q_V, Q_V)
Send	$IC_U = (Q_{CA}, ID_U, B_U, t_U) \rightarrow$	Receive
Receive	$\leftarrow IC_V = (Q_{CA}, ID_V, B_V, t_V)$	Send
Randomly choose k-bit integer c_U and 160-k bit integer r $d_U = h(c_U \| r) \in [2, n-2]$ Check validity of t_V $H(IC_V) \rightarrow e_V$ $R = (d_U e_V) \times B_V + d_U \times Q_{CA}$ $= d_U \times Q_V$ $D_U = d_U \times P$ $E = (D_U, (c_U \| r) \oplus R.x) = (E_1, e_2)$		Check validity of t_U $H(IC_U) \rightarrow e_U$ $Q_U = e_U B_U + Q_{CA}$
Send	$E = (E_1, e_2) \rightarrow$	Receive
		$q_V \times E_1 = R$ $u = e_2 \oplus R.x = c_U \| r$ Check if $E_1 = h(u) \times P$ If yes, get c_U as the most significant k bits of u. Choose a random k-bit integer c_V $y = E_{c_U}(ID_V \| c_V)$.
Receive	$\leftarrow y$	Send
$E_{c_U}^{-1}(ID_V \| c_V)$ $MacKey \| LinkKey$ $= KDF(c_U \| c_V \| ID_U \| ID_V)$ $z = q_U H(MacKey) + d_U \pmod n$		$MacKey \| LinkKey$ $= KDF(c_U \| c_V \| ID_U \| ID_V)$
Send	$z \rightarrow$	Receive
		$z \times P$ $= H(MacKey) \times Q_U + E_1$? $z' = MAC_{MacKey}(ID_V \| ID_U)$
Receive	$\leftarrow z'$	Send
$z' = MAC_{MacKey}(ID_V \| ID_U)$?		

Figure 2. Hybrid key establishment protocol.

calculated from genuine Q_V, provided that V later evidences knowledge of the corresponding private key q_V. U then encrypts c_U by using the provably secure elliptic cure encryption [9], and sends to V the ciphertext $E = (D_U, (c_U \| r) \oplus R.x) = (E_1, e_2)$, where $R.x$ is the x coordinate of R.

4. V decrypts the received message and obtains R by calculating $q_V \times E_1 = q_V d_U \times P = d_U \times Q_V = R$. V then computes $u = e_2 \oplus R.x$, and checks if $E_1 = H(u) \times P$. It yes, V obtains c_U as the most significant k bits of u. Otherwise, the protocol is terminated. V then selects a k-bit random number c_V as its link key contribution, and encrypts c_V concatenated with its identity ID_V using symmetric key encryption under key c_U, generating $y = E_{c_U}(ID_V \| c_V)$. Note that proper encryption mode needs to be used, such as the Cipher Block Chaining (CBC) mode, where the results of encrypting previous blocks affect the encryption of the current block. This ensures that there is no way for any device W to derive $E_{c_U}(c_V)$ from $E_{c_U}(ID_V \| c_V)$ and change this value.

5. V sends y to U. V also computes $MacKey \| LinkKey = KDF(c_U \| c_V \| ID_U \| ID_V)$, where KDF is the specified key derivation function, $LinkKey$ is the established link key, and $MacKey$ is for explicit key confirmation use. V then destroys c_U and c_V from its memory.

6. U decrypts the incoming message under c_U and checks if the decrypted message contains a proper coding of ID_V concatenated with some number. If the check fails, U terminates the protocol. Otherwise, U denotes the number as c_V, and U has verified that V has the knowledge of the private key q_V associated with Q_V. U computes $MacKey \| LinkKey = KDF(c_U \| c_V \| ID_U \| ID_V)$, and $z = q_U H(MacKey) + d_U \pmod n$. U then sends z to V and destroys c_U and c_V from its memory.

7. V verifies if $z \times P = h(MacKey) \times Q_U + E_1$. If it is false, V terminates the protocol. Otherwise, V believes that U has the knowledge of the private key q_U associated with Q_U, and U has provided the explicit key confirmation to V. V sends $z' = MAC_{MacKey}(ID_V \| ID_U)$ to U, where MAC is a message authentication code function.

8. U checks if z' is valid. If yes, V provides the explicit key confirmation to U and both sides take $LinkKey$ as the final established link key and accept the connection.

4. SECURITY ANALYSIS

Since we use the public key certification strategy to bind a device's public key with its unique identity, there is no need to protect and maintain a large online database for every device's secret key at KDC.

Authentication is accomplished by sending the challenge pairs (E, y) and (y, z). It is infeasible for an adversary to compute the correct response y without knowing q_V. Thus U can be sure that only V can produce the response and U verifies that V has the knowledge of the private key q_V associated with the certified Q_V. Also, $z \times P = H(MacKey) \times Q_U + E_1$ can be satisfied only if z is calculated by the correct private key q_U associated with the certified public key Q_U. Therefore, V can be sure that only U can produce the correct response. However, we should note that the symmetric key encryption $E_{c_U}(ID_V, c_V)$ must be properly implemented. For example, if it is implemented, e.g., in ECB mode or in stream cipher mode, then any adversary could derive

$E_{c_U}(c_V)$ from $E_{c_U}(ID_V, c_V)$, arbitrarily modify the value of $E_{c_U}(c_V)$ sent to U and leave $E_{c_U}(ID_V)$ unchanged. This can be avoided by using Cipher Block Chaining (CBC) mode. In addition, an adversary cannot obtain any information of c_U and c_V if both the symmetric and ECC encryption schemes are secure, which implies the link key contribution of each side is transferred securely to the other part.

We observe that both U and V supply their contributions c_U and c_V to the link key respectively. The final link key is established using the key generation function. In this way, no single party has the full control on the selection of the link key, and both U and V can ensure the freshness of the final link key.

To establish the link key, we include the names of the entities in the scope of the key derivation function to avoid the unknown key-share attack [23].

The protocol provides both implicit and explicit key confirmation. If U does not terminate the protocol in the 6th step, V knows c_U is received correctly and it can compute the link key correctly. If V does not terminate the protocol in the 8th step, U knows c_V is received correctly and it can compute the link key correctly. Therefore, if there is no "run with failure" message, both U and V confirm to each other implicitly that the key is generated correctly. Furthermore, we use *MacKey* to provide explicit key confirmation. If both z and z' authenticated by *MacKey* are verified as valid, then both U and V know that the other side actually has computed the link key correctly.

To reveal the sensor's static private key q_U, two sessions i and j with the same values of d_U and with the values of $H(MacKey_i), H(MacKey_j)$ known are necessary to an adversary. However, note that both c_U and r are chosen randomly, therefore, the ephemeral private key $d_U = H(c_U \| r) \in [2, n-2]$ is randomly chosen for each session. Since n is 160 bits large, the probability that $d_{Ui} = d_{Uj}$ is negligible for different sessions i and j. Since no single party has the full control on the selection of the link key, we also conclude that for $i \neq j$, $MacKey_i \neq MacKey_j$ hence $H(MacKey_i) \neq H(MacKey_j)$. If an adversary wants to obtain q_U, he needs to solve the following equations: $z_i = q_U H(MacKey_i) + d_{Ui} \pmod n \quad (0 < i < n)$ with unknown q_U and d_{Ui}. It is not easier than solving the elliptic-curve discrete logarithm problem.

If the security manager's private key is compromised, then all the link keys from earlier runs can be recovered from the transcripts. However, the corruption of the sensor node does not help to reveal the link keys. Therefore, our scheme provides half forward secrecy. To provide full forward secrecy, c_V should be sent to U in a secure way that only U with its ephemeral private key can reconstruct it. However, this requires additional expensive elliptic curve random point multiplications on sensor side, and is opposite to our purpose of offloading the computation burden of sensors. Since the sensor may be a weak device while the security manager can support much stronger security features, we believe that forward secrecy on the sensor side is more important than that on the security manager side.

5. PERFORMANCE ANALYSIS

The performance evaluation given here is based on the cryptographic operation complexity and the number of times they have to be performed, the sizes of the messages, the total number of messages sent in each protocol run and the memory requirement. Since sensors are much more battery and computational resources limited while the security manager is much more powerful, we restrict our attention to the efficiency of the sensor side only. Throughout this section, we mainly use the measurement results given by implementation of our protocol on the 16-bit single-chip microprocessor M16C [10] [17] [18] with 10 MHz clock rate designed by Mitsubishi Electric Corporation.

5.1 Computation Complexity

Note that the base point P and CA's public key Q_{CA} are fixed parameters. We can hence reduce the scalar multiplication of fixed points P and Q_{CA} by having a pre-computed look-up table in the ROM area. Since we verify the binding of the sensor's private key q_U to its public key Q_U in step 6 and 7 through a linear combination of the static key and the ephemeral key, rather than a multiplicative combination as in other ECC based protocols, at least one expensive elliptic-curve scalar multiplication of a random point is moved to the security manager side, and is replaced by one low cost modular multiplication, one modular addition and one symmetric key decryption. In real-time execution, the sensor is required to compute only one elliptic-curve scalar multiplication of a random point (B_V), two elliptic-curve scalar multiplication of fixed points (P and Q_{CA}), one symmetric key decryption, one modular multiplication, one modular addition, one hash, one key derivation and two random number generations.

On the average, SHA-1 only takes 2 msec to digest a 128-bit binary string on the M16C. Hash functions are also used to generate pseudo-random numbers. Thus their generation speed is comparable to that of hashing and can be ignored. The Rijndael [8] (AES) algorithm in Cipher Block Chaining mode takes less than three milliseconds to decrypt a 256-bit ciphertext. Therefore we can ignore the time taken for hashing, symmetric key operation, key derivation and random number generation in our evaluation.

The sensor also needs to do one 160-bit modular multiplication and modular addition, which takes less than 3 msec. Using Karatsuba-Ofman algorithm [15] and Montgomery product algorithm [20] to implement the 160-bit modular multiplication in 16-bit word fashion, the calculation has a complexity of (Appendix):

$$T_{Mon\,Pro}(160) + \frac{160}{16} Add(16) \approx 115.4 Mul(16) + 1640.3 Add(16),$$

where $Add(16)$ denotes the computation cost of an "addition" operation of two 16-bit numbers and $Mul(16)$ denotes that of "multiplication".

Therefore, the main computation cost at the sensor side is the elliptic-curve scalar multiplication of a random point and two fixed points, as can be shown by their computation complexity as:

$T_{EC-RP} \approx 123831.6Mul(16) + 1749703.9Add(16)$, and

$T_{EC-FP} \approx 24143.3Mul(16) + 341144.5Add(16)$ (Appendix).

The implementation result on M16C is 480 msec for a random point scalar multiplication and 130 msec for a fixed point multiplication. The whole protocol execution time on M16C is about 760 msec.

5.2 Communication Complexity

During the real-time execution of the hybrid protocol, a total of 6 massages are exchanged, two for mutual authentication and implicit certificates, two for the afterwards link key generation process and another two for the explicit key confirmation. If we assume the device ID is 64 bits, the certificate expiration time and the random number *k* are also 64 bits each, and the modulus for ECC is 160 bits, the total communication cost is 1437 bits or 180 bytes.

5.3 Code Space

The non-volatile (FLASH) memory required for a sensor to store its static public and private key pair (q_U, Q_U) and the implicit certificate $IC_U = (Q_{CA}, ID_U, B_U, t_U)$ is 96 bytes. The program memory (ROM) needed is a total size of 5.2K byte code/data (lots of room for optimizations), including 200 bytes for ECC implicit certificate generation and verification, 720 bytes of elliptic arithmetic library, 630 bytes modular *p* integer library, 790 bytes of general integer library, 410 bytes of SHA-1, 1K bytes of AES symmetric key algorithm and 1400 bytes of pre-computed data table, 20 byte base prime *p* and the 20 byte curve order *n*.

6. FURTHER ENHANCEMENT

We can further reduce the computation complexity on sensor side, by using the Modular Square Root (MSR) technique [26] to encrypt sensor's link key contribution c_U instead of using ECC cryptography. The attractiveness of MSR for wireless network application arises from its asymmetry. MSR requires the sending party to perform only a single modular multiplication, while the receiver performs exponentiation (needed to calculate the modular square root). Since our program includes the general integer library, it is easy to implement MSR operations using the library.

We call the modified protocol an MSR-combined hybrid key establishment protocol. First, an elliptic curve *E* defined over *GF*(*p*) (where *p* is the characteristic of the base field) with suitable coefficients and a base point *P* of large order *n* is selected and made public to all users. The sensor *U* randomly chooses integer $q_U \in [2, n-2]$ as its private key and computes the public key $Q_U = q_U \times P$. For the more powerful security manager *V*, we use N_V to denote the corresponding public key, i.e. the MSR modulus. $N_V = p_V q_V$, where p_V and q_V are large prime numbers. Since MSR is used at the security manager side, we use the elliptic curve digital signature algorithm (ECDSA) [1] as the signature scheme of CA.

In order to receive a certificate, the sensor sends its public key Q_U together with its user identity through an out-of-band secure interface to CA. CA uses its private key q_{CA} to sign the hashed value of the concatenation of the public key, the device identity ID_U , and the certification expiration date t_U . The CA then sends

U		CA
$q_U \in [2, n-2]$		$k_U \in [2, n-2]$
$Q_U = q_U \times P$		$R_U = k_U \times P$
Send	$Q_U, ID_U \rightarrow$	Receive
		$r_U = R_U.x \bmod n$
		$e_U = H(Q_U.x, ID_U, t_U)$
		$s_U = \frac{e_U + q_{CA} \cdot r_U}{k_U} \bmod n$
Receive, Store $q_U, Q_U, Q_{CA}, ID_U, (r_U, s_U), t_U$	$\leftarrow (r_U, s_U), Q_{CA}, t_U$	Send

Figure 3. ECDSA certificate generation for sensor *U*.

V		CA
$N_V = p_V q_V$		$k_V \in [2, n-2]$
		$R_V = k_V \times P$
Send	$N_V, ID_V \rightarrow$	Receive
		$r_V = R_V.x \bmod n$
		$e_V = H(N_V, ID_V, t_V)$
		$s_V = \frac{e_V + q_{CA} \cdot r_V}{k_V} \bmod n$
Receive, Store $N_V = p_V q_V, Q_{CA}, ID_V, (r_V, s_V), t_V$	$\leftarrow (r_V, s_V), Q_{CA}, t_V$	Send

Figure 4. ECDSA certificate generation for security manager *V*.

the signed message (r_U, s_U) together with its public key Q_{CA} through the secure channel to the terminal as shown in Figure 3. By repeating the very same process, the security manager *V* acquires its certificate (r_V, s_V) as shown in Figure 4. The certificate generation processes for sensor *U* and security manager *V* are performed offline and before they join the network. At the beginning of our MSR-combined hybrid key establishment protocol, they both send to the other side their public key, device ID, certificate and the expiration time. Then the mutual certificate authentication between the sensor and the security manager is executed in real-time, as shown in Figure 5.

U		V
Send	Q_U, ID_U, t_U $(r_U, s_U) \rightarrow$	Receive and check if t_U is valid
Receive and check if t_V is valid	$\leftarrow N_V, ID_V,$ $t_V, (r_V, s_V)$	Send
$c = s_V^{-1} \bmod n$		$c = s_U^{-1} \bmod n$
$e_V = H(N_V, ID_V, t_V)$		$e_U = H(Q_U.x, ID_U, t_U)$
$v_1 = c \cdot e_V \bmod n$		$u_1 = c \cdot e_U \bmod n$
$v_2 = c \cdot r_V \bmod n$		$u_2 = c \cdot r_U \bmod n$
$R = v_1 \times P + v_2 \times Q_{CA}$		$R = u_1 \times P + u_2 \times Q_{CA}$
if $R.x \neq r_V$, abort		if $R.x \neq r_U$, abort

Figure 5. Mutual authentication and certificate verification.

The MSR-combined hybrid key establishment protocol now proceeds as below and shown in Figure 6:

1. Both the sensor U and the security manager V send to the other side the public key, device ID together with the certificate. Then the mutual authentication and certificate verification is performed. If any check fails, the protocol is terminated.

2. U randomly selects a k-bit integer c_U, as its link key contribution, and encrypts it using V's public key N_V, generating $x = (r_U \| c_U)^2 \bmod N_V$ (r_U is the proper padding). U then randomly chooses an integer $d_U \in [2, n-2]$ and computes $D_U = d_U \times P$. (d_U, D_U) is used as U's ephemeral key pair.

3. U sends D_U and x to V.

4. V decrypts x and obtains c_U as the least significant k bits of $\sqrt{x} \bmod N_V$. V then selects a k-bit random number c_V as its link key contribution, and encrypts c_V concatenated with its identity ID_V under key c_U, generating $y = E_{c_U}(ID_V \| c_V)$. Note that proper encryption mode needs to be used, such as the Cipher Block Chaining (CBC) mode. V sends y to U.

5. V computes $MacKey \| LinkKey = KDF(c_U \| c_V \| ID_U \| ID_V)$, where KDF is the specified key derivation function, $LinkKey$ is the established link key, and $MacKey$ is for explicit key confirmation use. V then destroys c_U and c_V from its memory.

6. U decrypts the incoming message under c_U and checks if the decrypted message contains a proper coding of ID_V concatenated with some number. If the check fails, U terminates the protocol. Otherwise, U denotes the number as c_V, and U has verified that V has the knowledge of the private key associated with the certified public modulus N_V. U computes $MacKey \| LinkKey = KDF(c_U \| c_V \| ID_U \| ID_V)$, $z = q_U H(MacKey) + d_U \pmod n$. U then sends z to V and destroys c_U and c_V from its memory.

U	V
(d_U, Q_U)	$N_V = p_V q_V$
Mutual authentication and certificate verification	
Choose random k-bit c_U.	
$x = (r_U \| c_U)^2 \bmod N_V$.	
$d_U \in [2, n-2]$.	
$D_U = d_U \times P$.	
Send $x, D_U \rightarrow$	Receive
	$\sqrt{x} \bmod N_V \rightarrow c_U$, as the least significant k bits.
	Choose random k-bit c_V.
	$y = E_{c_U}(ID_V \| c_V)$.
Receive $\leftarrow y$	Send
$E_{c_U}^{-1}(ID_V \| c_V)$	$MacKey \| LinkKey$
$MacKey \| LinkKey$	$= KDF(c_U \| c_V \| ID_U \| ID_V)$
$= KDF(c_U \| c_V \| ID_U \| ID_V)$	
$z = q_U H(MacKey) + d_U \pmod n$	
Send $z \rightarrow$	Receive
	$z \times P$
	$= H(MacKey) \times Q_U + D_U$?
	$z' = MAC_{MacKey}(ID_V \| ID_U)$
Receive $\leftarrow z'$	Send
$z' = MAC_{MacKey}(ID_V \| ID_U)$?	

Figure 6. MSR-combined Hybrid key establishment.

7. V verifies if $z \times P = h(MacKey) \times Q_U + D_U$. If it is false, V terminates the protocol. Otherwise, V believes that U has the knowledge of the private key q_U associated with Q_U, and U has provided the explicit key confirmation to V. V sends $z' = MAC_{MacKey}(ID_V \| ID_U)$ to U, where MAC is a message authentication code function.

8. U checks if z' is valid. If yes, V provides the explicit key confirmation to U and both sides take $LinkKey$ as the final established link key and accept the connection.

Note that by using MSR to encrypt sensor's link key contribution c_U, only one 1024-modulus squaring is performed instead of doing the much more expensive random point elliptic-curve scalar multiplication in our previous scheme. The MSR encryption process comprises one modular addition and one modular multiplication and it takes only 45 msec to perform a 1024-bit MSR encryption. The computation complexity of using Karatsuba-Ofman algorithm [15] and Montgomery product algorithm [20] to implement the 1024-bit modular multiplication in 16-bit word fashion, is approximately (Appendix): $T_{Mon\,Pro}(1024) \approx 2187.3 Mul(16) + 30718.8 Add(16)$, which is much less than the complexity of elliptic-curve scalar multiplications.

Table 1. Comparison of the hybrid key establishment protocol and its MSR-combined version with other public-key based key establishment protocols

	Computation complexity on sensor side				Processing time on sensor	Communication complexity
	EC-RP	EC-FP	Large modular exponentiation	Small modular exponentiation		
Beller-C-Y			1	2	10.4 sec	4352 bits
Aziz-Diffie			2	3	20.4 sec	5120 bits
ECMQV X509	1.5	3			1110 msec	1796 bits
ECMQV implicit	2	1.5			1155 msec	1478 bits
ECDSA	2	3			1350 msec	1730 bits
ECDHE	2	3			1350 msec	1796 bits
Hybrid	1	2			760 msec	1437 bits
MSR-Hybrid		3		1	455 msec	3682 bits

In real-time execution of the MSR-combined hybrid key establishment protocol, the sensor is required to compute three elliptic-curve scalar multiplication of fixed points (two for verifying the ECDSA signature and another one for generating the ephemeral key), one symmetric key decryption, one modular multiplication, one modular squaring, one modular addition, one hash, one key derivation and two random number generations. The expensive public key decryption and elliptic-curve scalar multiplication of a random point are all moved to the security manager side, which is more computational powerful. The total processing time on M16C at the sensor side is approximately 455 msec.

However, the communication overhead is increased due to a larger key size used by the security manager. If we assume the device ID is 64 bits, the certificate expiration time and the random number k are also 64 bits each, and the modulus for ECC and Rabin cryptosystem are 160 bits and 1024 bits respectively, the total communication cost is 3682 bits or 460 bytes. If we assume the following battery drain: transmit-12.96 μJ/byte, receive-16.2 μJ/byte (based on Motorola figures on a 900 MHz transceiver), and 66 mW as the power consumption of M16C chip, the MSR-combined hybrid protocol saves 3.23 mJ on sensor side compared with executing our first hybrid key establishment protocol. However, if the message is sent multi-hops, the MSR-combined hybrid protocol consumes more energy at intermediate routers.

7. COMPARISON TO OTHER PUBLIC-KEY BASED PROTOCOLS

In this section, we compare our hybrid key establishment protocol and its MSR-combined version to other ECC based protocols, including the ECMQV protocol with ECC X509 certificates [27] and implicit certificates [7], the ECDSA authenticated key exchange protocol [1] running in ephemeral key mode and the Elliptic-Curve Diffie-Hellman Ephemeral (ECDHE) protocol [24]. We also compare our protocols to the Aziz-Diffie protocol [2] based on RSA (1024-bit key) and the Beller-Chang-Yacobi's protocol [4] based on Rabin cryptosystem (1024-bit key). We compare the sensor side computation complexity, processing time on M16C and the bandwidth requirements. We list the comparison results of our hybrid key establishment protocols with other public-key based key establishment protocols in Table 1, where computation complexity on sensor side is expressed in numbers of performing the cryptographic operations, including elliptic curve scalar multiplication of a random point (EC-RP) and a fixed point (EC-FP), modular exponentiation of a large number and that of a small number.

Table 1 shows that both our hybrid protocol and its MSR-combined version require less processing time hence less power consumption of computing the link key. The hybrid key establishment protocol also achieves the least bandwidth requirements, while its MSR-combined version has the least processing time but requires modest communication complexity compared with other public-key based key establishment protocols.

8. CONCLUSION

Sensors have rigid constraints on computational resources, processing power, and parameter storage space. In this paper, we propose a hybrid authenticated key-establishment protocol, in which we reduce the computation intensive elliptic curve scalar multiplication of a random point at the sensor side, and use symmetric key cryptographic operations instead. On the other hand, it authenticates the two identities based on elliptic curve implicit certificates, solves the key distribution and storage problems, which are typical bottlenecks in pure symmetric-key based protocols. The hybrid key establishment protocol has less sensor side computation complexity and less communication complexity as compared to other public-key based key establishment protocols. We also present an MSR-combined version of the hybrid key establishment protocol, which combines the use of MSR, ECC and symmetric cryptography. The MSR-combined hybrid protocol achieves the fast processing time on sensor side, while has a modest communication overhead.

9. REFERENCES

[1] M. Aydos, T. Yan and C. K. Koc. A High-speed ECC-based Wireless Authentication Protocol on an ARM Microprocessor. 16th Annual Computer Security Applications Conference (ACSAC'00), Dec. 2000, New Orleans, Louisiana.

[2] A. Aziz, and W. Diffie. A secure communications protocol to prevent unauthorized access - privacy and authentication for

wireless local area networks. IEEE Personal Communications, First Quarter (1994).

[3] S. Basagni, K. Herrin, E. Rosti and D. Bruschi. Secure pebblenets. Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001) (2001), 156-163.

[4] M. J. Beller, L.-F. Chang, and Y. Yacobi. Privacy and authentication on a portable communications system. IEEE Journal on Selected Areas in Communications, vol.11, no.6 (1993).

[5] Simon Blake-Wilson and Alfred Menezes. Authenticated Diffie-Hellman key agreement protocols. In 5th annual international workshop, SAC'98, 339-361.Springer-Verlag (1998).

[6] C. Boyd and A. Mathuria. Key establishment protocols for secure mobile communications: A selective survey. Proceeding of Australasian Conference on Information Security and Privacy (1998), 344-355.

[7] Certicom Research, Standard for efficient cryptography, SEC 1: Elliptic Curve Cryptography. Version 1.0, September 20, 2000. Certicom Corporation. URL: www.secg.org.

[8] J. Daemen and V. Rijmen. AES Proposal: Rijndael. AES Algorithm Submission, Sep 1999. http://www.nist.gov/aes.

[9] E. Fujisaki, T. Kobayashi, H. Morita, H. Oguro, T. Okamoto, S. Okazaki, and D. Pointcheval. PSEC: Provably secure elliptic curve encryption scheme. Primitive submitted to NESSIE by NTT Corp., September 2000.

[10] T. Hasegawa, J. Nakajima and M. Matsui. A small and fast software implementation of elliptic curve cryptosystems over GF(p) on a 16-bit microcomputer. IEICE Trans. Fundamentals, vol. E82-A, no.1 (1999).

[11] J.-P. Hubaux, L. Buttyan and S. Capkun. The quest for security in mobile ad hoc networks. Proceeding of ACM Symposium on Mobile Ad Hoc Networking and Computing (2001).

[12] IEEE P1363 Working Draft Appendices, Feb. 6, 1997.

[13] IEEE Std. 802.15.4-2003, IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements-Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs). New York: IEEE Press. 2003.

[14] M. Jakobsson and D. Pointcheval. Mutual authentication for low-power mobile devices. In Proceedings of Financial Cryptography 2001. Springer-Verlag (2001).

[15] D. E. Knuth. The art of computer programming: seminumerical algorithms. volume 2. Reading, MA: Addison-Wesley, Second edition (1981).

[16] C. K. Koc, T. Acar and B. S. Kaliski Jr. Analyzing and comparing montgomery multiplication algorithms. IEEE Micro, n.16, v.3, 26-33 (1996).

[17] User Manual of M16C/60 Series. Mitsubishi Electric Corporation, 1996.

[18] Software Manual of M16C/60 Series. Mitsubishi Electric Corporation, 1996.

[19] N. Modadugu, D. Boneh and M. Kim. Generating RSA keys on a handheld using an untrusted server. RSA 2000 (2000).

[20] P. L. Montgomery. Modular multiplication without trail division. Mathematics of Computation, vol.44, no.170, 519-521 (1985).

[21] A. Perrig, R. Szewczyk, V. Wen, D. Culler and D. Tygar. SPINS: Security protocols for sensor networks. Wireless Networks Journal (2002).

[22] R. Rivest, A. Shamir and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems, Communications of the ACM, vol. 21, 120-126 (1978).

[23] P. Rogaway, M. Bellare and D. Boneh. Evaluation of security level of cryptography: ECMQVS (from SEC 1), Jan. 2001.

[24] SECG, Elliptic Curve Cryptography, Standards for Efficient Cryptography Group, 2000. Available from http://www.secg.org/collateral/sec1.pdf.

[25] V. Shoup. Practical threshold signatures. Advances in Cryptology, Eurocrypt'00 (2000), 207-220.

[26] D.R. Stinson. Section 4.7 The Rabin Cryptosystem, Cryptography: Theory and Practice, CRC Press (1995).

[27] Rene Struik and Gregg Rasor, "Mandatory ECC Security Algorithm Suite", submissions to IEEE P802.15 Wireless Personal Area Networks, March 2002.

[28] D. S. Wang and A. H. Chan. Mutual authentication and key exchange for low power wireless communications. In IEEE MILCOM 2001 Conference Proceedings (2001).

[29] D. S. Wang and A. H. Chan. Efficient and mutually authenticated key exchange for low power computing devices. In Asiacrypt'01, LNCS. Springer-Verlag (2001).

[30] L. Zhou and Z. J. Hass. Secure ad hoc networks. IEEE Network Magazine, vol. 13, no. 6 (1999), 24-30.

APPENDIX: COMPLEXITY ANALYSIS OF MODULAR EXPONENTIATION AND ELLPTIC CURVE MULTIPLICATION

The Montgomery method [20] is an efficient way for modular multiplication with an arbitrary modulus. Assuming the modulus n is an N-bit number, let r be 2^N. The Montgomery algorithm transforms an integer m in the range $[0, n-1]$ to another integer in the same range, which is called the image or the n-residue of the integer, and is defined as $\overline{m} = mr \bmod n$. It is easy to show that the Montgomery multiplication over the images $\overline{a}$ and $\overline{b}$ computes the image $\overline{c} = \overline{a}\overline{b}r^{-1} \bmod n$ which is the same as the integer $c = ab \bmod n$ [16]. Assume a, b and n are N-bit integers. Let n' be the integer so that $rr^{-1} - nn' = 1$. The average

complexity of Montgomery product algorithm, which computes $\bar{c} = \bar{a}\bar{b}r^{-1} \bmod n = ab \bmod n$, is given below:

Function MonPro($\bar{a}, \bar{b}$)
Step 1. $t := \bar{a} \cdot \bar{b}$ // $Mul(N)$
Step 2. $u := t \cdot n' \bmod r$ // $Mul(N)$
Step 3. $\bar{c} := (t + u \cdot n)/r$
// $Mul(N) + 2Shift(2N, N) + Add(N)$
Step 4. if $\bar{c} > n$ then return $\bar{c} - n$
else return $\bar{c}$ // $condition + \frac{1}{2}Add(N)$

Here $Shift(x, y)$ means the computation cost of shifting an x-bit integer by y bits. $Add(N)$ denotes the computation cost of an "addition" operation of two N-bit numbers and $Mul(N)$ denotes that of "multiplication". The average complexity of Montgomery product algorithm is approximately:

$$T_{Mon\,Pro}(N) \approx 3Mul(N) + \frac{3}{2}Add(N).$$

In practice, primitive arithmetic operations such as multiplication and addition are limited to a certain word size k. Karatsuba-Ofman algorithm [15] is an efficient way to perform the $N * N$ bit multiplication in k bit fashion, which keeps on dividing the long integer into two shorter ones of equal sizes until their lengths are k, and gets the multiplication of two long integers by doing multiplications and additions on their divided parts of the half length. Let a_1 and a_0 denote the higher and the lower halves of a, respectively, and b_1, b_0 denote the higher and the lower halves of b. Karatsuba-Ofman algorithm is as follows:

KORMA (a, b)
Step 1. if (a and b are of more than $2k$ bits) do
Step 2. $t_0 := \text{KORMA}\ (a_0, b_0)$ // $T_{KO}(N/2)$
Step 3. $t_2 := \text{KORMA}\ (a_1, b_1)$ // $T_{KO}(N/2)$
Step 4. $u_0 :=$ KORMA $(a_0 + a_1, b_0 + b_1)$
// $T_{KO}(N/2+1) + 2Add(N/2)$
Step 5. $t_1 := u_0 - t_0 - t_2$ // $2Add(N)$
Step 6. else do
Step 7. $t_0 := a_0 \cdot b_0$
Step 8. $t_2 := a_1 \cdot b_1$
Step 9. $u_0 := (a_1 + a_0) \cdot (b_1 + b_0)$
Step 10. $t_1 := u_0 - t_0 - t_2$
Step 11. return $(2^N t_2 + 2^{N/2} t_1 + t_0)$
// $Shift(N, N) + Shift(N, \frac{N}{2}) + Add(2N, N)$

$T_{KO}(N)$ denotes the arithmetic/logic operations needed for $N * N$ bit multiplication, and $T_{KO}(N) \approx 2T_{KO}(\frac{N}{2}) + T_{KO}(\frac{N}{2}+1) + 2Add(\frac{N}{2}) + 2Add(N) + Add(2N, N) \approx 2T_{KO}(\frac{N}{2}) + T_{KO}(\frac{N}{2}+1) + 5Add(N)$. If we implement the $(\frac{N}{2}+1)*(\frac{N}{2}+1)$ bit multiplication by:

$$a(\frac{N}{2}+1)*b(\frac{N}{2}+1) = \{a'*2^{\frac{N}{2}} + a(\frac{N}{2})\}*\{b'*2^{\frac{N}{2}} + b(\frac{N}{2})\} =$$
$$a(\frac{N}{2})*b(\frac{N}{2}) + a'*b(\frac{N}{2})*2^{\frac{N}{2}} + b'*a(\frac{N}{2})*2^{\frac{N}{2}} + a'*b'*2^N,$$

where a' and b' are the highest bit of $a(\frac{N}{2}+1)$ and $b(\frac{N}{2}+1)$, and if the recurrence stops when a and b are k bits, we have

$$T_{KO}(N) \approx 3^{\log_2(\frac{N}{k})} Mul(k) + 7\sum_{i=1}^{\log_2(\frac{N}{k})} 3^{i-1} Add(\frac{N}{2^{i-1}}).$$

Assuming $Add(N) = 2Add(\frac{N}{2})$, then we conclude that using Karatsuba-Ofman algorithm to implement the $N * N$ multiplication on k-bit processors, the average complexity of Montgomery product algorithm to compute $c = ab \bmod n$ (where a, b and n are N-bit integers) is:

$$T_{Mon\,Pro}(N) \approx 3[(\frac{N}{k})^{\log_2 3} Mul(k) + 14(\frac{N}{k})^{\log_2 3} Add(k)] + \frac{3N}{2k}Add(k).$$

The average computation cost of the binary scalar multiplication algorithm of a random point is $T_{EC-RP} \approx \frac{160-1}{2}[T_{EC-doub} + \frac{1}{2} \cdot T_{EC-add}]$, where $T_{EC-doub}$ denotes the computation complexity of doing an elliptic curve point doubling and T_{EC-add} denotes that of doing an elliptic curve point addition. Note that the base point P is a fixed system parameter. We can hence reduce the scalar multiplication of the base point by having a pre-computed look-up table in the ROM area. The standard window method is adopted for the fixed point multiplication and the average computation cost of this window scalar multiplication of a fixed point is $T_{EC-FP} \approx \frac{31}{2}[T_{EC-doub} + \frac{1}{2} \cdot T_{EC-add}]$.

The IEEE-P1363 document [12] describes a detailed implementation algorithm that realizes the elliptic addition and doubling, and we can estimate the computation complexity by the number of modular multiplications. The elliptic addition normally involves 11 modular multiplications, except at the last step of ECDSA verification, where 16 modular multiplications are needed. The elliptic doubling can be performed by 8 modular multiplications. Therefore, we can readily estimate that

$$T_{EC\ addition} \approx 33[(\frac{160}{k})^{\log_2 3} Mul(k) + 14(\frac{160}{k})^{\log_2 3} Add(k)] + \frac{2640}{k}Add(k);$$
$$T_{EC-doub} \approx 24[(\frac{160}{k})^{\log_2 3} Mul(k) + 14(\frac{160}{k})^{\log_2 3} Add(k)] + \frac{1920}{k}Add(k).$$

Hence, the average computation complexity of the random point scalar multiplication is approximately:

$$T_{EC-RP} \approx \frac{12879}{4}[(\frac{160}{k})^{\log_2 3} Mul(k) + 14(\frac{160}{k})^{\log_2 3} Add(k) + \frac{80}{k}Add(k)].$$

The average computation complexity of a fixed point scalar multiplication is approximately:

$$T_{EC-FP} \approx \frac{2511}{4}[(\frac{160}{k})^{\log_2 3} Mul(k) + 14(\frac{160}{k})^{\log_2 3} Add(k) + \frac{80}{k}Add(k)].$$