Effective Key Establishment and Authentication Protocol for Wireless Sensor Networks Using Elliptic Curve Cryptography

K.S.B.DEEPTHI¹, UPPE NANAJI² AND Y.SWAPNA³

¹² St.Theressa College Of Engineering,Dept of CSE, Garividi, VZM(Dt), nanajiet@gmail.com, deepthi.kakaraparthi@gmail.com
³ vignal college of engg, Dept of CSE, Garividi, VZM(Dt), swapna.yellabilli@gmail.com

ABSTRACT: In this paper, we propose an efficient authenticated key establishment protocols between a sensor and a security manager in a self-organizing sensor network and also 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 simulated using MATLAB, 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. The proposed protocol has lower storage requirements for the user side, which makes it suitable for smartcards and other handheld computing devices.

Keywords— Elliptic Curve Cryptography, Wireless Sensor Networks.

INTRODUCTION

In a Wireless Sensor Network (WSN for short), individual sensor nodes, or sensors, are constrained in energy, computing, and communication capabilities. Typically, sensors are mass-produced anonymous commodity devices that are initially unaware of their location. Once deployed, sensors should self-organize into a network that works unattended. Due to the fact that individual sensor nodes are anonymous and that communication among sensors is via wireless links, sensor networks are highly vulnerable to security attacks. The task of securing WSNs is an open research problem. A solution must strike a tradeoff between the security provided and the consumption of energy, computing, and communication resources in the nodes. One way to implement secure wireless communications in WSNs is through the use of message encryption. In the simplest case, a number of nodes in the network share a secret encryption key(s), henceforth called session key(s).

SCALABLE KEY ESTABLISHMENT

In this section, we propose three different key establishment protocols for Wireless Sensor networks according to different application scenarios, the first one based on pure symmetric key cryptography and suitable for toys/games home networking, the second based on a hybrid of symmetric key and public key cryptography and designed for residential or small commercial wireless network deployment, and the last one based on pure public key cryptographic operations and targeting large industrial or military applications.

Perrig et al. present to use trusted third parties to assist node-to-node key agreement. We call this trusted third party a security manager. A security manager is granted special capabilities to assist in provisioning link keys to other end mobile devices on-site. The security manager should first establish a link key with an end device before it can install link keys into that device for secure communicating with other end devices inside the mobile cluster. One way to accomplish the initial link key establishment task is to pre-install a master key table into each device. However, Wireless Sensor Networks may be highly versatile, involving temporary communications between devices that may have never met before. Thus we cannot predict and install all master keys needed for devices before they join the network, especially for large-scale wireless sensor networks. An alternative way is to use a shared group key that is pre-loaded into each device in our proposed cryptosystem. Then when two devices want to establish a link key, they use this group key to encrypt and exchange their ephemeral key contribution data. Since the group key is fixed, the trust relationship is based merely on the knowledge of the other device’s extended IEEE 64-bit address. The computation complexity and power consumption of symmetric key cryptographic operations are negligible when compared with public key schemes. However, a common group key poses a security risk if any one device is compromised. Therefore, this pure symmetric key based key establishment protocol in applications that
require the least security protection, such as the home
toys/games automation.

The use of asymmetric keys along with digital
certificates to establish individual link keys can help reduce
this risk and restrict the impact of key compromise to the
compromised node itself, rather than to all its key-sharing
parties. Public-key operations are quite expensive though.
In recent years, ECC based key agreement protocols have
gained popularity in constrained mobile environments, due
to the property of small key sizes. We proposed a hybrid
key establishment protocol, which is based on a
combination of ECC and symmetric key operations. The
motivation is to exploit the difference in capabilities
between security managers and end devices, and put the
cryptographic burden where the resources are less
constrained. End mobile devices 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 end device side and replaces them
with low cost and efficient symmetric key based
operations.

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. In our mobile cryptosystem, we use the
elliptic curve implicit certificate scheme [4], 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
$qCA$ as its static private key, and computes the static public
key $QCA = qCA . P$.

To obtain a certificate and the static private-public key
pair, an end device $U$ randomly selects a temporary key
pair $(gU, G_U)$ and sends $G_U$ to CA. CA verifies $U$’s identity
and the authenticity of the request received from $U$. CA
also selects a temporary key pair $(gV, G_V)$ and computes
the elliptic curve point $B_V = G_V + G_U$. The implicit
certificate $ICV$ for $U$ is constructed as the concatenation of
CA’s static public key $QCA$, the device identity $ID_U$, the
elliptic curve point $B_V$, and the certificate expiration date
$ICV_0$, i.e., $ICV = (QCA, ID_U, B_V, ICV_0)$. CA then applies a one-way
hash function $H$ on $ICV$ and derives an integer $x \in [2, n-2]$ from
$H(ICV)$ following the conversion routine. Finally, CA
computes $U$’s private-key reconstruction data $s_U = g_U x +
qCA (mod n)$. CA’s public key $QCA = eB_V + QCA$, and sends $s_U$ and
$ICV$ back to $U$. After $U$ obtains the implicit certificate
from CA, it computes the hash value $H(ICV)$ and derives an
integer $x$ from $H(ICV)$ following the conversion routine. $U$
also computes its static private key $qCA = sU + gU x + eU (mod n)$
and its public key $Q = qCA . P$. $U$ then reconstructs the
public key $Q = eB_V + QCA$. If $Q = qCA . U$ accepts the
certificate and outputs the static key pair $(qCA, QCA)$;
otherwise it rejects the certificate. By repeating the very
same process, a security manager $V$ acquires its certificate
$ICV$ (and static key pair $(qV, QV)$).

The certificate generation processes for end device $U$
and security manager $V$ are performed offline and before
they join the network. When they first communicate to each
other, they execute our hybrid key establishment protocol
as below:

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 $H(ICV)$ and derives an
   integer $eV$ from $H(ICV)$ following the conversion
   routine. $V$ then obtains $U$’s public key $Q = eB_V + QCA$.
   After performing the certificate processing, $V$ can
   conclude that $Q$ is genuine, provided that $V$ later
   evidences knowledge of the corresponding private key
   $qCA$.

3. $U$ selects a k-bit random number $cU$ as its link key
   contribution and a random 160-bit integer $r$. $U$
calculates its ephemeral private key $dU = H(cU || r)$ and
   ephemeral public key $D_U = dU . P$, where $H$ is a
cryptographic hash function to map a binary string to a
   random integer $E \in [2, n-2]$. $U$ verifies $V$’s certificate and
   obtains $V$’s public key the same way as $V$ does, but
   instead of computing $Q$ directly, $U$ computes $R = dV$
   $sV = (dV . x ) B + dV x QCA$. $U$ can conclude that $R$
is calculated from genuine $Q$, provided that $V$
later evidences knowledge of the corresponding private key
$QCA$. $U$ then encrypts $cU$ by using the provably secure
elliptic curve encryption [6], and sends to
   $VE = (D_U (cU || V || y))$. $V$

4. $V$ decrypts the received message and obtains $R$ by
   calculating $qCA . x E = qCA . dV . x P = dV . x QCA = R$. $V$
   then computes $u = eV . R . x$, and checks if
   $E = H(u) . x P$. It
   yes, $V$ gets $cU$ as the most significant $k$
   bit of $u$. Otherwise, the protocol is terminated.
   $V$ then selects a k-bit random number $cV$, as its link key contribution, and
   encrypts $cV$ concatenated with its identity $ID_U$ using
   symmetric key encryption under key $cU$, generating
   $y = E(cU || ID_U || cV)$. Note that proper encryption mode needs
   to be used, such as the Cipher Block Chaining (CBC)
   mode, which ensures that there is no way for any device
   $W$ to derive $E(cV)$ from $E(ID_U || cV)$ and change this value. $V$ sends $y$
to $U$.

5. $V$ computes $MacKey || LinkKey = KDF(cV || ID_U || ID_V)$, 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 $cV$ and $cU$ from its memory.
6. \( U \) decrypts the incoming message under \( c_v \) and checks if the decrypted message contains a proper coding of \( ID_u \) 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 \( q_5 \) associated with \( Q_v \). \( U \) computes \( \text{MacKey} \) by \( \text{LinkKey} = KDF(c_v||ID_u||ID_v) \), and \( z = q_5H(\text{MacKey}) + d_1 (\text{mod } n) \). \( U \) then sends \( z \) to \( V \) and destroys \( c_v \) from its memory.

7. \( V \) verifies if \( x_5 = h(\text{MacKey})Q_v + E \). If it is false, \( V \) terminates the protocol. Otherwise, \( V \) believes that \( U \) has the knowledge of the private key \( q_5 \) associated with \( Q_v \), and \( U \) has provided the explicit key confirmation to \( V \). \( V \) sends \( z' = MAC(\text{MacKey}, ID_u, ID_v) \) 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 \text{LinkKey} as the final established link key and accept the connection.

In this protocol, 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 \( q_5 \). Thus \( U \) can be sure that only \( V \) can produce the response and \( U \) verifies that \( V \) has the knowledge of the private key \( q_5 \) associated with the certified \( Q_v \). Also, \( z' x P = H(\text{MacKey})Q_v + E \), can be satisfied only if \( z' \) is calculated by the correct private key \( d_1 U \) associated with the certified public key \( Q_v \). Therefore, \( V \) can be sure that only \( U \) can produce the correct response. In addition, an adversary cannot obtain any information of \( c_v \) 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.

This hybrid key establishment protocol consumes more node energy as compared to the pure symmetric key based protocol. However, 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 public key 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. Therefore, our hybrid key establishment protocol is faster and saves more node energy than other public key based protocols, as evidenced by running our protocol using MATLAB. The whole protocol execution time on end device side is about 760 msec, while ECMQV protocol with ECC X509 certificates and implicit certificates takes 1110 msec and 1155 msec respectively, and the Elliptic-Curve Diffie-Hellman Ephemeral (ECDH) protocol takes 1350 msec.

The hybrid key establishment protocol has much better security enhancement than our first pure symmetric key based protocol, while has moderate energy consumption on end mobile devices. We notice that 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 and is suitable to use in residential and small commercial mobile applications where security is important but not critical, and we can trade security for mobile users’ energy efficiency.

To provide full forward secrecy, rather than being encrypted under a symmetric key \( c_v \), \( c_v \) should be sent to \( U \) in a similar way that \( c_v \) is sent to \( V \) (i.e., through secure elliptic-curve encryption [6]), and only \( U \) with its ephemeral private key can reconstruct it. Then our hybrid protocol is modified into a pure ECC based public-key key establishment protocol.

However, this requires additional expensive elliptic curve random point multiplications on mobile user side, and is opposite to our purpose of offloading the computation burden of end devices. The pure ECC based public-key key establishment protocol is suitable to vital or security-sensitive network deployments, including natural disaster control, battlefield rescue, rescue missions, etc., where security is more important than energy efficiency.

**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 \( c_v \) 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 as 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 \( q_u \) [2,n-2] as its private key and computes the public key \( Q_u \) more powerful security manager \( V \), we use \( N_v \) to denote the corresponding public key, i.e., the MSR modulus. \( N_v = p.q_u \), where \( p \) and \( q_u \) are large prime numbers. Since MSR is used at the security manager side, we use the elliptic curve digital signature algorithm (ECDSA) 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 \( q_v \) to sign the hashed value of the concatenation of the public key, the device identity \( ID_v \) and the certification...
expiration date \( t_u \). The CA then sends the signed message \((r_u, s_u)\) together with its public key \(Q_u\), through the secure channel to the terminal. By repeating the very same process, the security manager \(V\) acquires its certificate \((r_v, s_v)\). 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.

The MSR -combined hybrid key establishment protocol now proceeds as below:

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 \(N_u\), generating \(x = (r_u || ctu) \mod N(vru)\) is the proper padding). \(U\) then randomly chooses an integer \(d_U \in \mathbb{Z}_{[2, n^2]}\) and computes \(D_U = d_uP\cdot(d_U, D_U)\) is used as \(U\)’s ephemeral key pair.

3. \(U\) sends \(D_U\) and \(x\) to \(V\).

4. \(V\) decrypts \(x\) and obtains \(cu\) as the least significant \(k\) bits of \(ntx \mod Nv\). \(V\) then selects a \(k\)-bit random number \(c_v\) concatenated with its identity \(ID_v\) under key \(c_{uv}\) generating \(y = E_v(ID_v, c_v)\). 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 \(c_v\) and checks if the decrypted message contains a proper coding of \(ID_u\), concatenated with some number. If the check fails, \(U\) terminates the protocol. Otherwise, \(U\) denotes the number as \(c_u\), and \(U\) has verified that \(V\) has the knowledge of the private key associated with the certified public modulus \(N_u\). \(U\) computes \(MacKey \parallel Link\) \(Key = KDF(c_u || c_v || ID_u || ID_v)\), \(z = q_uH(MacKey) + d_u \mod n\), \(U\) then sends \(z\) to \(V\) and destroys \(c_u\) and \(c_v\) from its memory.

7. \(V\) verifies if \(z \cdot x = p\cdot H(MacKey) \cdot s_u + D_u\). 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 \(c^*_v\), 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 bits to perform a 1024-bit MSR encryption which is much less than the complexity of elliptic-curve scalar multiplications.

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 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. 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.

**SIMULATION RESULTS**

The user and the server have to get their certificates from the third party which will be the certifying authority. This process refers to implicit certificate generation process, finally authentication and key establishment process occurs.

**Implicit Certificate Generation Process**

Elliptic curve implicit certificate is used to avoid the typical key management problem in pure symmetric-key based protocols. The elliptic curve implicit certificate scheme are used, because of the resulting low communication complexity, which is a dominant factor for low bit transmission channels in sensor networks as shown in Fig. 1. This generation process takes processing time of merely 1115 ms for 1500 bits of key size.
RSA Based Aziz-Diffie Protocol

The Aziz-Diffie protocol is used when the user contacts a server over the vulnerable “air interface” of a typical sensor networks. Aziz-Diffie protocol uses signature authentication. It assures mutual assurance of key freshness to prevent replays of old messages being used to re-establish an “old”, possibly compromised, session key. Fig. 2 shows that processing time of Aziz-Diffie protocol is 20.4 sec for merely 5120 bits of key size.

Mutual Authentication and Key Agreement Protocol

Mutual authentication and key agreement protocol is used to establish an agreed key and session key between the user and security manager of sensor network which provides the authentication to encrypt/decrypt the message. Fig. 3 shows that, for 1730 key size it will take processing time of 1350 ms to finish the authentication process.

Proposed Hybrid Key Establishment Protocol

Simulation results shows that both Hybrid protocol require less processing time of 1350 ms hence less power consumption of computing the link key. The hybrid key establishment protocol also achieves the least bandwidth requirements, as shown in Fig. 4.

In real-time execution, the sensor is required to compute only one elliptic-curve scalar multiplication of a random point, two elliptic-curve scalar multiplication of fixed points, one symmetric key decryption, one modular multiplication, one modular addition, one hash, one key derivation and two random number generations. Further it requires only 760 ms to process these protocol for key size 1437 bits.

Modified MSR-combined hybrid key establishment Protocol

Reduction of computation complexity on sensor side is done, by using the Modular Square Root (MSR) technique to encrypt sensor’s link key contribution 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).

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 number 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 at the sensor side is approximately 455 msec.

Fig. 5: Processing time versus key size for MSR-c ombined hybrid key Establishment protocol

5. COMPARISON OF SIMULATION RESULTS
Table 1 show 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.

Table 1: Comparison of the different protocol

<table>
<thead>
<tr>
<th>Protocols</th>
<th>Processing time</th>
<th>Key Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aziz-Diffie</td>
<td>20.4 sec</td>
<td>5120 bits</td>
</tr>
<tr>
<td>ECMOV implicit</td>
<td>1155 msec</td>
<td>1478 bits</td>
</tr>
<tr>
<td>ECDSA</td>
<td>1350 msec</td>
<td>1750 bits</td>
</tr>
<tr>
<td>ECDHE</td>
<td>1350 msec</td>
<td>1796 bits</td>
</tr>
<tr>
<td>Hybird</td>
<td>760 msec</td>
<td>1437 bits</td>
</tr>
<tr>
<td>MSR-Hybrid</td>
<td>455 msec</td>
<td>3682 bits</td>
</tr>
</tbody>
</table>

While its MSR-combined version has the least processing time but requires modest communication complexity compared with other public-key based key establishment protocols.

K.S.B.Deepthi is currently pursuing her post graduation course as a computer science engineer from JNTUK, Avanthi’s St.Theressa college of engg. Garividi, VZM.

CONCLUSIONS
Sensors have rigid constraints on computational resources, processing power, and parameter storage space. In this paper, we propose a hybrid authenticated key-establishment protocol, i 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.

REFERENCES

Uppa Nanaji. HOD CSE & IT DEPT, working in St.Theressa college of Engg, GARIVIDI, VZM(Dt),A.P,INDIA. Having 12 reaching experience. Published 12 international journals. Uppa Nanaji doing his PhD from University of Allahabad in C.S.E. Areas of interests is wireless Networks, Computer Networks and Sensor Networks.