JOURNAL OF THEORETICAL PHYSICS & CRYPTOGRAPHY

www.IJTPC.org/Publications

Feasibility of Public Key Cryptography in Wireless Sensor Networks

Murari Mandal, Gaurav Sharma, Suman Bala, Anil K Verma
Thapar University, Patiala-147004, India

ABSTRACT

Wireless Sensor networks can employ sensor nodes numbering from few hundreds to thousands. For such large number of sensor nodes communicating with each other, there is need for providing security for various reasons. Although Symmetric key cryptography is favoured for being efficient but it becomes impractical for handling such large number of entities. Public key cryptography (PKC) like ECC, RSA, and pairing based cryptography can be employed to handle it. In this paper we are discussing about the implementation of PKC in wireless sensor networks. Our discussion includes the corresponding setup for such implementations, exploring the various decisions that have to be taken while selecting the PKC schemes and the parameters for such schemes.

Keywords

PKC, wireless sensor networks (WSN), ECC, RSA, AVRORA, TinyOS, RELIC, bilinear map.

1. INTRODUCTION

Wireless Sensor networks (WSN) [1] are composed of sensor nodes. The number of such nodes may vary from few hundred to thousands or even more. In WSN, a sensor node is generally assigned with the task of sensing some data, which can be of various types such as temperature, pressure, speed, voice, direction, images etc. Various kinds of sensors are available to carry out these tasks like acoustic sensors, thermal sensors, magnetic sensors, seismic sensors etc. A sensor node must be able to perform some local computation based on the sensed data, which requires memory space and processing capabilities. In order to transmit its data a node has to communicate with other nodes. For all these activities a node is dependent on the power source, which is limited and in most cases it is a battery, which is irreplaceable. So life of a sensor node becomes directly proportional to the life of its battery in most cases.

The application of WSN in an environment where the nodes are accessible to any kind of outsider makes the network insecure. So security in such sensor networks is a vital issue that must be addressed properly. The security issues are tampering with transmitted data among the nodes, unauthorized access of data from any node, introducing an unauthorized node in the network, listening to the communication among the nodes etc. To address all these security issues a proper cryptographic protocol or scheme must be employed in the sensor network. The choice of such a scheme depends on the constraints put up by the nature of WSNs. Primary constraints on a node are the limited computational capabilities, limited memory size available and limited power resources. As a WSN, there are further requirements such as scalability, fault tolerance and handling the topology change in the network that has to be fulfilled. Also it may not be always possible to have global ID for all the nodes in the network.

Symmetric key cryptography [2] although requires less computational time however can't help in achieving all the security aspects for the WSNs. Also management of such large number of nodes (and keys corresponding to them) becomes impractical. Public key cryptography [3] on the other hand does consume more computational time but it provides the desired security requirements. For last one decade, the area of implementing security in wireless sensor networks [4] has been of great interest for the researchers and many PKC schemes has been suggested and implemented. In this paper, we are presenting an overall discussion about the implementation of PKC in WSNs, the tools required, choosing the parameters for the PKC schemes and their limitations and advantages and various cryptographic standards.

The rest of the paper is organized as follows: Section 2 describes the limitations of symmetric key cryptography. In Sections 3, we present the experimental Setup required for the implementation of PKC in WSNs. In section 4, we present

the decisions regarding the PKC schemes followed by the conclusion in Section 5.

2. LIMITATIONS OF SYMMETRIC KEY CRYPTOGRAPHY

In Symmetric key cryptography [2], the secret key is shared between the sender and the receiver. Sender uses the key k to perform the encryption and after receiving the encrypted message the receiver decrypts using the same key. In a WSN, it is not advisable to give the same key to all the nodes. Compromise of any one node can lead to compromise of security for the entire network. An alternate approach has been provided in literature to distribute random key pool to the sensor nodes. Following are the reasons for not preferring this scheme in wireless sensor networks:

- *Key Connectivity:* Keys are allotted to the sensor nodes before deployment. The probability of a node communicating with its neighbors depends on the common keys in the pool of keys allotted to it, so communication establishment is not always a sure event.
- *Resilience/ Fault tolerance:* A wireless sensor network is prone to the failures of nodes due to reasons such as power insufficiency, external attacks. This may affect the symmetric key cryptographic scheme heavily since every node has a pool of keys. A single node capture can lose a pool of keys.
- *Scalability*: Topology of a WSN may change, more nodes may be added. The protocol can't handle the dynamic nature of the WSN properly.
- *Key Revocation*: When some node is compromised then the collection of keys it was having must be revoked immediately, otherwise the attacker may use them to infiltrate the network. This removal of keys will affect the neighbor nodes.
- *Communication cost*: Symmetric cryptosystem accounts for more communication cost compared to the PKC. It will affect the battery life of the sensor node, which is very precious in WSN.

3. EXPERIMENTAL SETUP

3.1 TinyOS Programming

TinyOS [5] is a component-based operating system specifically designed for WSN, in collaboration between the University of California, Berkeley, Intel Research and Cross-bow Technology. TinyOS is an open software system. The aim of TinyOS is to supportthe concurrency intensive operations required by WSN with minimal hardware requirements. It is implemented in nesC (network embedded system C) programming language, which is a dialect of C language. The additive tools have a support of Java and shell script front-ends like TOSSIM, which are having a support of python. Other associated libraries and tools like nesC compiler and AVR binutilstoolchains, are written in C language. The constructions of TinyOS programs are basically of software components, which are connected to each other through various interfaces.

In nesC all the components are working in local namespace. This is the main difference between nesC and other languages like C, C++ and Java. Let component X has declared that it calls a function A, so X is bringing the name $X \cdot A$ into the global namespace. Another component Y might be calling function A but it introduces $Y \cdot A$ into the global name space. So, both X and Y are calling to the function but the implementation of $X \cdot A$ and $Y \cdot A$ might be totally different.

- *Components*: In a component, we declare the functions it provides (implements) and the function it uses (calls). Interfaces are used instead of directly declaring each and every function (of both kinds).
- *Interfaces*: Interfaces are collection functions. Related Functions are declared in the interfaces and the components have to just specify the interfaces it wants to use.
- *Wiring*: Wiring is nothing but connecting the components and connecting providers and users together. *provides* command implies the function the component is implementing and other components can take their service. *uses* command implies that the function is implemented by some other component.
- *Configurations*: There are two kinds of components modules and configurations. Implementations are done in module and configurations describe how the modules are wired together. A configuration may also include other configurations and wiring among them. Configurations also perform the task of exporting interfaces. Exporting is nothing but mapping one name to another, which is useful in many cases. Three operators are used $\rightarrow$,$\leftarrow$ and $=$. First two are used for wiring and the third one $=$ is used for exporting interfaces (also called pass-through wiring).

Since, in WSNs, the direct user interaction with the nodes or the embedded devices is minimum as compared to the general PC usages, so the static approach in nesC is justified. The wiring occurs at compile time, which minimizes the RAM usage at runtime. With TinyOS, it has become possible to write code with minimum memory storage requirements and also with low power consumption, which are the essential requirements of WSN.

3.2 Choice of Library

Writing algorithms for the cryptographic operations is not only time consuming but also there is fair chance that the efficiency may not be up to the mark. Various libraries are available (both open source and proprietary) who provide the algorithms for efficient implementations of cryptographic operations. MIRACL and RELIC are the two libraries, which have full support for RSA, ECC and pairing, based cryptography. RELIC is an open source library, which has been developed by and is being maintained by UNICAMP and MIRACL is from Shamus Software. In the related literature, these two libraries have been preferred more than other libraries for the implementation of PKC.

- ***MIRACL*:** The algorithms in MIRACL (Multiprecision Integer and Rational Arithmetic C/C++ Library) [6] works on 18 characters sized blocks. Other than RSA and ECC (for both prime fields and binary fields) it also supports Diffie-Hellman Key exchange and DSA digital signature. Though very popular among cryptographers, MIRACL has not been used much in the embedded systems where space constraint is very high. MIRACL is the most efficient library for 80x86/Pentium platform.

- ***RELIC*:**The primary focus of RELIC [7] is efficiency and flexibility. Especially for ECC implementation, RELIC is more often the first choice for the researchers and developers. In RELIC, algorithms work on blocks of 40 characters. Another important feature is to provide help in building architecture dependent code. RELIC provides a complete set of algorithms for multi-precision integer arithmetic, ECC, Bilinear maps (Tate pairing and optimal pairing), extension fields, RSA, BLS short signatures, ID-based authenticated key agreement, Rabin, ECMQV and ECSS.

Pigatto et al. [8] compared two algorithms ECC and El-Gamal for MIRACL and RELIC libraries and concluded that RELIC outperforms MIRACL for parameters like response time and key-size. The comparison between the average response times achieved by ECC based algorithms is performed using message size of 50KB and considering the two different key sizes 160-bit and 256-bits. The algorithm based on MIRACL library has a considerably higher time than the one based on RELIC, in both cases. The times obtained are approximately 9 seconds in case of MIRACL and 3.3 seconds in case of RELIC, in which the key size is 160-bit. When using 256-bit key size, the times are 20.9 seconds in case of MIRACL and 10.6 seconds in case of RELIC.

3.3 AVRORA

AVRORA [9] is a cycle-accurate simulator designed for sensor networks and is written in Java. A Compilers group in UCLA has been working on this project. It is intended to simulate and analyze the programs written for AVR microcontrollers, MICAz and MICA2 motes. With the help of AVRORA, simulation can be performed on sensor networks of thousands nodes. There is no GUI support for AVRORA but it does a good job in consuming less execution time. It provides a framework for program analysis. The memory consumption of the program code can be computed by compiling the program using command *"make micaz"*. After the compilation AVRORA is used to calculate the number of cycles, the commands are written as *"convert-avrora main.exe main.od"*, it will convert the *.exe* file to *.od* file. The command *"avrora -platform=micaz -monitors=sleep main.od"* will count the number of cycles in active and sleep state, by using this count, running time and energy consumption can be calculated easily. The formula for energy consumption is as follows [10]:

Total Energy Consumption = Voltage Level × Running Time

4. DECISIONS REGARDING PKC SCHEMES

In PKC, one public key and one private key is used. The scheme relies on the intractability of the problem of deriving the private key from the corresponding public key. There have been many PKC schemes suggested. RSA, ElGamal public-key encryption and signature schemes and its variants and the ECC are most commonly used. RSA derives its security from the hardness of IFP (Integer Factorization problem), ElGamal and variants from the DLP (Discrete Logarithm Problem) and ECC from the ECDLP (Elliptic curve discrete logarithm problem). Following are some of the decisions, which have to be taken while implementing a PKC scheme in WSN.

4.1 Selection of Pairing

For last few years (since, the paper by Joux [11] in 1985 on implementing a cryptographic protocol using bilinear pairing) a lot of research is being carried out for designing and implementing cryptographic protocols using pairing. In EllipticCurve Cryptography, the implementation of bilinear pairing can be done using Tate Pairing or Weil pairing. Many cryptographic protocols that are based on bilinear pairing depend on the intractability of Bilinear Diffie-Hellman Problem (BDHP) for security. There are many protocols suggested based on the application of pairings like three-party one-round key agreement by Joux [11] as modified by Verheul [12], short signatures [13], identity based encryption [14], short group signature scheme [15] etc.

While implementing cryptographic protocols for WSN, a large number of keys have to be generated and managed. Traditionally, certificates are used for these purposes, which are generated by Certificate Authority (CA). A certificate (lets say for Alice) contains identification information and public key of Alice and also contains the signature of CA on the data. Due to practical difficulties in managing certificates, IBE (Identity Based Encryption) can be a preferred option. A practical identity based encryption protocol was suggested first by Boneh and Franklin [16] using Weil pairing. Weil pairing or Tate pairing on elliptic curves can be used to implement an IBE scheme.

Let an elliptic curve E is defined over the field F_q. The order of E over F_q is $\#E(F_q) = nh$, where n is a prime number and h is small integer (e.g. $h = 2,3,4$). There exists a small positive integer d such that$n|q^d - 1$. The set of all points $P \in E(F_q)$ satisfying $nP = \infty$ is denoted by $E[n]$, where $E[n] \subseteq E(F_{q^d})$ [17].

- Weil Pairing:Let $P, Q \in E[n]$ be two points and D_p be some divisor, where $D_p = (P) - (\infty)$ (∞ is the identity of $E(F_q)$). Let nD_p is a principal divisor (since $nD_p = n(P) - n(\infty)$) [18] and f_p be a function with $(f_p) = nD_p$. Similarly, D_q and f_q can be defined. The Weil Pairing of P and Q is definedas [19]: $\hat{e}(P, Q) = \frac{f_p(D_q)}{f_q(D_p)}$, where $f_q(D_p) \neq 0$. Weil pairing is well defined and satisfies all the conditions of bilinear mapping.
- Tate Pairing:Let $P, Q \in E[n]$be two points. Let G_2 denote the order-n subgroup of $F_{q^d}^*$. The (modified) Tate Pairing is given by map $\hat{e}: E[n] \times E[n] \to G_2$, where $n \nmid q - 1, d > 1, gcd(n, h) = 1, n \nmid \#E(F_{q^d})/n^2$. Tate pairing can be defined as [18]: $\hat{e}(P, Q) = f_p(D_q)^{(q^d-1)/n}$.

The Tate pairing can be modified to be well defined, bilinear, non-degenerate and efficiently computable. Both Tate and Weil pairing can be computed with the help of algorithm [20]. A sound comparison between Weil pairing and Tate Pairing for implementing an IBE system is given by Oliveira and Aranha [21]. Tate pairing is favored with less computation time required than for Weil pairing except for some exceptional cases. The selection of pairing depends on the parameters for the particular cryptographic protocols. More details about bilinear pairing can be found in [18].

4.2 Choice of Field (Prime vs. Binary fields)

In ECC, we are primarily concerned with the field arithmetic and the elliptic curve arithmetic defined over a particular

field. An elliptic curve can be defined over a binary field (F_{2^m}) or a prime field (F_p), where the values of m and p are chosen according to the FIPS (Federal Information Processing Standards) [22] recommended fields to ensure the security of the protocols and the curves are chosen so that computation of reduction algorithms is fast.

- Prime field: The simplified Weierstrassβ equation for prime field is given by: $y^2 = x^3 + ax + b$, where $a = -3$. The point doubling in Jacobian coordinates can be computed with fewer multiplications. In the prime field, the use of Jacobian coordinates takes $4M$ (multiplication),$4S$ (squaring) for doubling and the uses of mixed Jacobian affine coordinates take $8M$, $3S$ for the addition of two points [23].
- Binary field: The simplified Weierstrassβ equation for binary field is given by: $y^2 + xy = x^3 + ax^2 + b$. The usage of Lopez-Daheb projective coordinates takes$4M$ for doubling and mixed coordinates (Lopez-Daheb and affine coordinates) take $8M$ for addition of two points [23].

The execution time in ECC schemes is dominated by the point or scalar multiplication operation. The point multiplication is defined as to compute kP, where k is a randomly generated integer and P is a point on an elliptic curve defined over a field. It can be concluded that binary field operations are less expensive than prime fields for comparable bit size. However, it is not always straightforward. For both prime and binary field, there exist various methods and algorithms to reduce the scalar multiplication time and required memory storage. The common examples are: double and add, sliding window method, wNAF method and Montgomery Ladder [24, 25].

In WSN, 8/16-bit processors are in common use. Wenger and Hutter [26] compared the performance of two distinct ECC-hardware implementations that are based on prime field (NIST P-192) and binary field (ANSI X9.62 c2tnb191v1) arithmetic and concluded that the F_{2^m} based processors outperform the F_p based processors in run time, area required to make the chip and power consumption.

4.3 Selection of Curve

NIST (National Institute of Standards and Technology), USA, has recommended a collection of elliptic curves for Federal government use and it contains choices of private key length and underlying fields.

The parameter sets of the binary koblitz curve standardized by NIST are listed below. The curves are of the form $y^2 + xy = x^3 + ax^2 + b$ over a binary field. For the curve *K*–163 the following parameters are listed as:

$p(t)$: the reduction polynomial (in explicit and hexadecimal form)

a: the curve's coefficient

G_x, G_y: the x and y coordinates of the base point G

n: the base point's order

h: the curve's cofactor

For K–163:

$$p(t) = \mathrm{t}^{163} + \mathrm{t}^7 + \mathrm{t}^6 + \mathrm{t}^3 + 1$$

$$= 800000000000000000000000000000000000000C9a$$

$$= 1$$

$$G_x = 2\mathrm{fe}13\mathrm{c}0537\mathrm{bbc}11\mathrm{acaa}07\mathrm{d}793\mathrm{de}4\mathrm{e}6\mathrm{d}5\mathrm{e}5\mathrm{c}94\mathrm{eee}8 G_y$$

$$= 289070\mathrm{fb}05\mathrm{d}38\mathrm{ff}58321\mathrm{f}2\mathrm{e}800536\mathrm{d}538\mathrm{ccdaa}3\mathrm{d}9$$

$$n = 5846006549323611672814741753598448348329118574063$$

$$h = 2$$

NIST has presented five "security levels" of 80, 112, 128, 192 and 256 bits, and a curve would match that level only if its key size is at least twice the security level. So the standard curve for each level ought to be the smallest curve, which is large enough for that level. This should yield Koblitz curves in fields of size 163, 233, 277, 409 and 571 bits, respectively.

4.4 Selection of key-size

In cryptography, an algorithm is said to be having *k*-bit security if key used in the algorithm is *k*-bits long and there is no computational attack possible on the algorithm. The security of an algorithm can never exceed the key length. In most symmetric key algorithms, security level is equal to key length but there are few exceptions also. For example, Triple DES provides at most 112 bits of security but key size used here is 168 bits. The less security is due to the existing computational attack of complexity 2^{112} known. There is no public key algorithm having the same level of security as key size; ECC provides the effective security of roughly half its key length. In WSN, 80-bit security is considered enough, which can be achieved using around 160-bit key in public key algorithms. For ECC implementation, curve *K*–163 over binary field has been chosen.

4.5 Cryptographic Standards

Cryptographic standards provide the facility for the use of proper cryptographic techniques and also it ensures the scope for interoperability among the various implementations. The standards are specified for the cryptographic schemes independent of the area of implementations. It provides the assurance to both the developers and the customers about the security. These standards have been specified by organizations that have thoroughly analyzed the parameters (to the date), the drawbacks and the loopholes (if present) in corresponding algorithms and protocols. These standards are verified and updated regularly. The standards specify several things like the parameters for the scheme, data formats, key size, steps for the schemes and the message exchanges. For providing security in WSN the PKC can follow either of the recommended standards.

- Standards for ECC: ANSI X9.62, ANSI X9.63, FIPS 186.2, SEC 1 and SEC 2 are some of the most popular standards followed in the literature. Other standards are IEEE 1363-2000, IEEE P1363a, ISO/IEC 15946-1, ISO/IEC 15946- 2 etc. [22].
- Standards for RSA: For the operations like generating RSA integers, key- generations and algorithms in RSA schemes various standards have been set. Some of the standards are FIPS 186-3, ANSIX9.44, ISO/IEC18033- 2, PKCS#1, IEEE 1363-2000, NESSIE etc. Loebenberger and Nusken[27] have analyzed the various standards for RSA and concluded that the various specifications do not affect the cryptographic properties of the generated integers significantly.

5. Conclusion

Choosing or designing an efficient protocol for implementing public key cryptographic protocol in wireless sensor networks is a topic, which is being explored by researchers all over the world. Many factors influence the selection or design of such a protocol like choosing the elliptic curve (in case of ECC), choosing an efficient pairing method and the modifications required, choosing the proper field, the order of the field, library for the algorithms, key sizes etc. There can be many different combinations of choices depending on the hardware characteristics, power constraints, computation time requirements and the memory space constraints. Depending on the characteristic of the WSN and the level of security that one wants to have, these decisions have to be made. We have provided an overall picture, which will help the readers to make such decisions.

6. REFERENCES

[1] Akyildiz I. F., Su W., Sankarasubramaniam Y., Cayirci E., 2002. Wireless sensor networks: A survey, Computer Networks, 38(4):393–422.

[2] Zhang X., Heys H., Cheng L., 2010. Energy efficiency of symmetric key cryptographic algorithms in wireless sensor networks, in 25th Biennial Symposium on Communications, QBSC '10, 168–172.

[3] Wander A., Gura N., Eberle H., Gupta V.,Shantz S., 2005. Energy analysis of public-key cryptography for wireless sensor networks, in proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications, PERCOM '05, 324–328.

[4] Chen X., Makki K., Yen K.,Pissinou N., 2009. Sensor network security: A survey, IEEE Communications Surveys Tutorials, 11(2):52–73.

[5] Levis P., Madden S., Polastre J., Szewczyk R., Whitehouse K., Woo A., Gay D., Hill J., Welsh M., Brewer E., Culler D., 2005.TinyOS: An operating system for sensor networks, Ambient Intelligence, 115–148.

[6] Shamus software ltd., MIRACL library. http://www.compapp.dcu.ie/~mike/ shamus.html.

[7] Aranha D.,Gouvˆea C. RELIC is an Efficient LIbrary for Cryptography. http: //code.google.com/p/relic-toolkit/.

[8] Pigatto D., Silva N.,Branco K., 2011. Performance evaluation and comparison of algorithms for elliptic curve cryptography with el-gamal based on MIRACL and RELIC libraries, Journal of Applied Computing Research, 1(2):95–103.

[9] Titzer B, Lee D., Palsberg J., 2005.AVRORA: scalable sensor network simulation with precise timing,in proceedings of the 4th International Symposium on Information Processing in Sensor Networks, IPSN '05, 477–482.

[10] Szczechowiak P., Oliveira L., Scott M., Collier M., Dahab R., 2008.NanoECC: Testing the limits of elliptic curve cryptography in sensor networks,Wireless Sensor Networks, LNCS, 4913:305–320.

[11] Joux A., 2004. A one round protocol for tripartite diffie-hellman, Journal of Cryptology, 17(4):263–276.

[12] Verheul E., 2004. Evidence that xtr is more secure than supersingular elliptic curve cryptosystems, Journal of Cryptology, 17(4):277–296.

[13] Boneh D., Lynn B.,Shacham H., 2004. Short signatures from the weil pairing, Journal of Cryptology, 17(4):297–319.

[14] Shamir A., 1985. Identity-based cryptosystems and signature schemes, Advances in Cryptology, LNCS,196 : 47– 53.

[15] Boneh D., Boyen X., Shacham H., 2004. Short group signatures, Advances in Cryptology CRYPTO 2004, LNCS, 3152: 41–55.

[16] Boneh D., Franklin M., 2001. Identity-based encryption from the weil pairing, Advances in Cryptology CRYPTO 2001, LNCS, 2139: 213–229.

[17] Balasubramanian R., Koblitz N., 1998. The improbability that an elliptic curve has subexponential discrete log problem under the menezesokamotovanstone algorithm, Journal of Cryptology, 11(2):141–145.

[18] Menezes A., 2009. An introduction to pairing-based cryptography. http://cacr.uwaterloo.ca/~ajmeneze/publications/pairings.pdf.

[19] Pippal R., Jaidhar C., Tapaswi S., 2012. Security vulnerabilities of user authentication scheme using smart card,in DBSec '12, 106–113.

[20] Miller V., 2004. The weil pairing, and its efficient calculation, Journal of Cryptology, 17(4):235–261.

[21] Oliveira L., Aranha D., Gouvˆea C., Scott M., Cˆamara D., L´opez J., Dahab R., 2011.Tinypbc: Pairings for authenticated identity-based non-interactive key distribution in sensor networks. Computer Communications, 34(3): 485–493.

[22] Johnson D., Menezes A., Vanstone S., 2010. The elliptic curve digital signature algorithm (ecdsa). http://cs.ucsb.edu/~koc/ccs130h/notes/ecdsa-cert.pdf.

[23] Hankerson D., Menezes A., Vanstone S., 2003. Guide to Elliptic Curve Cryptography. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[24] Gura N., Patel A., Wander A., Eberle H., Shantz S., 2004. Comparing elliptic curve cryptography and RSA on 8-bit CPUs, Cryptographic Hardware and Embedded Systems - CHES 2004, LNCS, 3156: 119–132.

[25] Liu A.,Ning P., 2008.TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks, in Proceedings of the 7th International Conference on Information Processing in Sensor Networks, IPSN '08, 245–256, Washington, DC, USA.

[26] Wenger E.,Hutter M., 2012. Exploring the design space of prime field vs. binary field ECC-hardware implementations, Information Security Technology for Applications, LNCS, 7161: 256– 271.

[27] Loebenberger D.,Nusken M., 2011. Analyzing standards for RSA integers, Progress in Cryptology, AFRICACRYPT 2011, LNCS, 6737: 260–277.