

Public Key Based Cryptoschemes for Data Concealment in Wireless Sensor Networks

Einar Mykletun
Computer Science Department
University of California, Irvine
mykletun@ics.uci.edu

Joao Girao and Dirk Westhoff
NEC Europe Ltd.
Kurfürsten-Anlage 36
69115 Heidelberg, Germany
{joao.girao,dirk.westhoff}@netlab.nec.de

Abstract—In-network data aggregation is a popular technique for reducing the energy consumption tied to data transmission in a multi-hop wireless sensor network. However, data aggregation in untrusted or even hostile environments becomes problematic when end-to-end privacy between sensors and the sink is desired. In this paper we revisit and investigate the applicability of additively homomorphic public-key encryption algorithms for certain classes of wireless sensor networks. Finally, we provide recommendations for selecting the most suitable public key schemes for different topologies and wireless sensor network scenarios.

I. INTRODUCTION

Wireless sensor networks (WSNs) are becoming increasingly popular in many spheres of life. Application domains include monitoring of the environment (such as temperature, humidity, entity movement and seismic activity) as well as numerous other ecological, law enforcement and military settings. Due to the limited amount of power a sensor is deployed with, the technique of data aggregation is commonly employed in an effort to minimize the amount of data that needs to be transmitted. It is a method which consists in condensing the sensed values according to a specific application and does not aim at reconstructing all the values at the receiver. By using data aggregation and in-network processing, the network can converge to a single result, such as, for example, the average, variance, minimum or maximum of the sensed values. This method can also be applied in certain points of the network, that store the values sensed over a region and time. Since applications often do not require every individual aggregated value, we can also aggregate for long-term storage, thus minimizing the amount of storage space required.

In this work we consider data privacy issues between readers, aggregators and the sensors themselves. We aim at providing end-to-end encryption of sensors' measurements as they are aggregated in the network via hop-by-hop transmission. We attempt to achieve the highest level of security possible, while taking power consumption and platform specific limitations as the major metric for the overall system. We outline a list of desired criteria that reflect the security and system requirements, and contrast a set of candidate solutions. We take the step towards the application of additive asymmetric privacy homomorphisms as a feasible solution to the problem of end-to-end aggregated encryption in some classes of WSNs.

To this end, we focus also on long-term data storage policies in WSNs, where we aim at a higher security than previous work [9], since data needs to be persistently stored at the nodes. The main contribution of our work is to give recommendations for the selection of the preferred scheme when attempting to ensure data privacy and aggregation in WSNs.

II. RELATED WORK

The problem of aggregating encrypted data in WSNs was introduced by Girao et al. in [9] and further refined in [3]. The authors propose to use homomorphic encryption schemes to enable arithmetic operations over ciphertexts that need to be transmitted in a multi-hop manner. Domingo-Ferrer's symmetric cryptosystem [6], which is both additively and multiplicatively homomorphic, is applied in [9]. Although interesting from an application standpoint, it has restricted usability in WSNs, due to its weak security level, as it is limited to protect against ciphertext-only attacks. This might not be reasonable in settings where the monitored value domain may be relatively easy to determine (if sensors are measuring the temperature, an attacker can enumerate all possible measurement values) or if long-term storage is required. The authors in [3] propose to use a trivial variant of the one-time pad encryption scheme. Although this scheme is provably secure, it has the drawback that the aggregating sink node needs to know exactly which sensors were involved in the data aggregation process, in order to regenerate the key-streams required for decryption.

Common to all work related to secure aggregation is the absence of solutions involving public-key encryption schemes. However, public-key based solutions provide a higher level of system security, since nodes would not be equipped with private keys, which would limit the advantage gained by an attacker compromising some of the nodes. It is therefore important to note why public-key encryption schemes have not yet been deployed. The reason is two-fold: (1) they are deemed too costly for computationally weak devices and (2) the expansion in bit size during the transformation of plaintext to ciphertext introduces costly communication overhead, which directly translates to a faster depletion of the sensor's energy. In this paper we try to relax the statements above by investigating various suitable public-key schemes. We provide a comparison of their costs as well as indications as to how

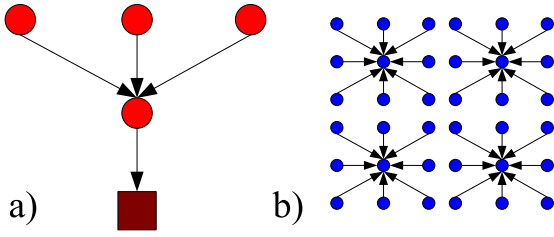


Fig. 1. Predominant data traffic flows in WSNs: a) synchronous and b) asynchronous.

practical they are. We build upon the work by Gura et al. in [10], who have demonstrated the feasibility of efficiently computing elliptic curve operations on the microcontroller of the Mica2 sensor nodes [1], and show that there indeed exists a viable public-key cryptosystem candidate for WSNs.

III. NETWORK AND ATTACKER MODEL

We consider a set of $S_i \in \mathcal{S}$ to be the sensing nodes in the network, a set of $A_i \in \mathcal{A}$ to be aggregator nodes, a set of $F_i \in \mathcal{F}$ as forwarders and R as the entity which requests the aggregate values, which we call the *reader*. Each S_i senses his environment and converts the reading into a value s_i . As these values travel hop-by-hop in a reverse multicast towards R , they are combined through aggregate functions such as $sum = \sum_{i=1}^n s_i$. The roles assigned to the nodes may vary in time as a result of protocols running in parallel, such as the Low Energy Adaptive Clustering Hierarchy [21], which try to flatten the overall amount of energy available in the network. This way, the set of $S_i \in \mathcal{S}$, $A_i \in \mathcal{A}$ and $F_i \in \mathcal{F}$ may change over time and their intersection may not always be \emptyset .

We further specify this model to distinguish between networks which are the direct application of a *flat* network of sensors, as described above, or a *layered*, hierarchical version which may require reading the actual aggregated value at intermediate nodes in the network, e.g. an intermediate post processing of aggregated data. Another characteristic of these networks is the time span for which the values need to be stored and the frequency at which they are sensed and aggregated. These factors impact the amount of data that can be aggregated at a single point in the network.

We can further classify sensor networks with respect to the supported traffic classes: *synchronous sensor networks*, where the focus is on the real-time nature of the readings and promptness of the results, and *asynchronous sensor networks*, where the values requested have a relatively small time constraint in the time since they are requested to when they need to be presented, as depicted in Fig. 1.

This paper stresses data concealment: our goal is to prevent passive and active attackers from learning aggregated values of the WSN. An attacker is assumed to be limited to a region, in the sense that it can monitor any value in this fraction of the network. Furthermore, we assume an attacker can gain physical access to individual sensors and read their internal state and values. Since we focus on the particular problem

of concealment, this approach should be used in combination with an authentication protocol to prevent data manipulation.

Since we expect a WSN to be deployed in an uncontrolled and unsupervised environment, and with the nodes vulnerable to being physically captured, the commonly used Dolev-Yao threat model [5] no longer covers all possible attacks. One possible remedy would be to equip sensor nodes with tamper resistant hardware units, but this comes at a cost that for a major set of applications we deem too expensive in terms of production or just too impractical in deployment. We argue that wireless sensor devices should be cheap enough to become ubiquitous units.

IV. A DESIRABLE HOMOMORPHIC CRYPTOScheme

In Section II we discussed previously proposed solutions to the problem of concealed data aggregation in WSNs and concluded that none were fulfilling all requirements. Prior to looking for alternative methods, we will outline the criteria and desired requirements of an optimal solution:

Aggregation:

- *Additively Homomorphic Encryption*: we wish to perform in-network data aggregation over encrypted data in order to reduce the amount of transmitted data. For this we require encryption schemes that are said to be *additively homomorphic*. They have the property that $Enc(m_1) \oplus Enc(m_2) = Enc(m_1 + m_2)$, where $Enc(m)$ represents the encryption of plaintext m and \oplus is an additive operation performed over ciphertexts.

Security:

- *Provable Security*: the security level of the encryption scheme should be measurable and it should be based upon the commonly agreed hardness of a mathematical problem to be provably computationally secure¹.

- *Sensor Compromise*: the compromise of a subset of sensor nodes should not assist in revealing aggregated data. No sensitive key material should be present at nodes.

- *System Security*: from the two points mentioned above, we can define the overall system security as being the weakest of the two.

- *Key Management*: the key management should be kept simple enough to avoid bandwidth intensive techniques needed to identify the encryption keys being used by sensors.

- *Ciphertext Expansion*: the expansion in bit size attributed to encryption should be moderate (not to lose the performance gain over hop-by-hop encryption).

- *Probabilistic Encryption*: encryption of the same plaintext should not, with high probability, yield the same ciphertext.

WSN Lifetime:

- *Efficient Computations*: cryptographic operations performed at sensors should not be overly expensive (as they significantly impact battery lifetime).

¹A cryptoscheme is said to be computationally secure if the cost of an attack outweighs the value of the encrypted data.

- *Bandwidth*: the bandwidth overhead attributed to sending ciphertexts should not require the transmission of large amounts of additional data.

- *Aggregator Node Election*: the algorithm for electing aggregator nodes should not need to take into account security parameters, thereby allowing it to make selections based upon the remaining energy level of the nodes.

The second *security* criterion rules out a hop-by-hop encryption approach, as the compromise of a few nodes may be enough to render the WSN insecure. While the third point reveals the weaknesses of symmetric key schemes in WSN settings when assuming non-tamper resistant sensors. Probabilistic encryption proves useful to avoid divulging information from ciphertexts only, as identical environment values may often be measured and encrypted by neighboring nodes. We note that non-tamper resistant sensor nodes can be compromised and have their contents revealed by an attacker (such as public-keys and current un-encrypted measurements). However, it should not be possible to learn (encrypted) aggregated values from the compromise of a single node or a minor fraction of the WSN. The *lifetime* criteria relate to the lifetime of nodes, as computations and especially communication are energy intensive. By performing in-network aggregation, nodes avoid having to forward every received packet towards the reader, thereby drastically reducing the overall bandwidth consumption.

Public Key Cryptoscheme Requirements: In light of the discussion above, we are encouraged to re-visit the use of public key encryption schemes that (1) are additively homomorphic (allowing for in-network aggregation of particular aggregation functions), (2) exert the required security levels, (3) involve relatively cheap computations, (4) are probabilistic, (5) produce relatively short ciphertexts, and (6) by nature of public key methodology require no sensitive key material to be stored at encrypting sensors. By fulfilling these requirements to some degree, a cryptoscheme would meet the desired properties put forth in this section. We are especially interested in the use of elliptic curve cryptoschemes, due to (a) their use of small keys which lead to short ciphertexts, (b) the smaller real-estate required for hardware implementations (number of gates) and (c) a better security-per-bit ratio.

V. PUBLIC-KEY CRYPTOScheme CANDIDATES

In this section we examine potential cryptoscheme candidates that meet some or all the desired criteria outlined in Section IV. In the remainder of the paper we denote encryption and decryption by $Enc()$ and $Dec()$, respectively.

In Eurocrypt 98', Okamoto and Uchiyama proposed a new public-key cryptosystem as secure as factoring and based on the ability of computing discrete logarithms in a particular sub-group [19]. Specifically, for an odd prime p , the p -Sylow subgroup is defined as $\gamma_p = \{x < p^2 \mid x = 1 \pmod{p}\}$, and $|\gamma_p| = p$. A function L that maps elements from γ_p to \mathbb{Z}_p is defined as $L(x) = (x - 1)/p$. Function L has homomorphic properties from multiplication to addition. For elements $a, b \in$

γ_p , $L(a * b) = L(a) + L(b) \pmod{p}$, and for $c \in \mathbb{Z}_p$, $L(a^c) = c * L(a)$.

Their scheme is characterized by probabilistic encryption, additive homomorphic properties, and relating the computational complexity of the encryption function to the size of the plaintext. We now describe their cryptosystem:

Let p and q be random k -bit primes and set $n = p^2q$. For an n of approximately 1024 bits, a choice of k could be 341. Next, randomly choose a $g \in_R \mathbb{Z}_n$ such that element $g_p = g^{p-1} \pmod{p^2}$ has order p . Finally, set $h = g^n \pmod{n}$. The additive homomorphic property is achieved through the multiplication of ciphertexts: $Enc(m_1 + m_2) = Enc(m_1) \times Enc(m_2)$.

Okamoto-Uchiyama (OU)	
Public Key	$n = p^2q, g, h$
Private Key	(p, q)
Encryption	plaintext $m \in 2^k$, $r \in_R \mathbb{Z}_n$, ciphertext $c = g^m h^r \pmod{n}$
Decryption	$c' = c^{p-1} \pmod{p^2}$ compute $m = L(c')L(g_p)^{-1} \pmod{p}$
Note that $c^{p-1} \pmod{p^2} = g^{m(p-1)} g^{nr(p-1)} = g_p^m \pmod{p^2}$	

In [12] Benaloh introduced a probabilistic cryptoscheme whose encryption cost is dependent on the size of the plaintext. The key-setup is as follows: let $n = pq$ for large primes p, q and choose value r such that $r \mid (p-1)$, $\gcd(\frac{p-1}{r}, r) = 1$ and $\gcd(q-1, r) = 1$. The public key y is chosen such that $y \in \mathbb{Z}_n^*$ and $y^{(p-1)(q-1)/r} \pmod{n} \neq 1$. The scheme's security is based upon the cryptographic assumption that it is computationally difficult to decide higher residuosity: given z, r and n of unknown factorization, find x such that $z = x^r \pmod{n}$. The additive homomorphic property is achieved through the multiplication of ciphertexts.

Benaloh	
Public Key	$n = pq, y, r$
Private Key	(p, q)
Encryption	plaintext $m \in \mathbb{Z}_r$, $u \in_R \mathbb{Z}_n^*$, ciphertext $c = y^m u^r \pmod{n}$
Decryption	compute m such that $(y^{-m'} c \pmod{n}) \in Enc_r(0)$ for $m' = 0, 1, 2, \dots$ until $r-1$ or $m = m'$

To understand why decryption works, it is useful to notice that the decryptor needs the ability to decide higher residuosity, which can be done efficiently when the factorization of n is known. Note that $z \in Enc(0)$ iff $z^{(p-1)(q-1)/r} \pmod{n} = 1$. Therefore, one can decrypt a ciphertext c by finding, via brute force, the smallest integer $m' < r$ such that $y^{-m'} c \pmod{n} \in Enc(0)$.

One method to speed up decryption is to store pre-computed values $T_i = y^{i(p-1)(q-1)/r} \pmod{n}$, for $i = 0, 1, \dots, r-1$ in a lookup table. Then, for $c = Enc(m)$, it is the case that $c^{(p-1)(q-1)/r} \pmod{n} = T_m$ and one can therefore avoid the brute force search by using the lookup table. For large values

of r it may be too expensive to store all r T_m values, and one can then resort to a *big-step little-step* method by only pre-computing T_i for $i \approx k\sqrt{r}$ as k ranges from 1 to r . Such an optimization reduces the storage, pre-computation time and decryption time to $O(\sqrt{r})$ at the decryptor.

In [17] Paillier describes three new probabilistic encryption schemes that use elliptic curves over rings, and exhibit additive homomorphic properties. All three are elliptic curve variants of previously described public key encryption algorithms, namely those proposed by Naccache-Stern [4], Okamoto-Uchiyama [19] and Paillier [16]. Since the encryption schemes in [17] are implemented over elliptic curves and meet some of our desired criteria, we describe each and investigate their applicability for aggregation in WSNs. Common to each scheme is that the elliptic curve is defined over \mathbb{Z}_n or \mathbb{Z}_n^2 , where n is the product of large primes, and that they are provably secure against chosen plaintext attacks. All three schemes provide additive homomorphic capabilities through the summation of ciphertexts. The reader is referred to [17] for more detailed descriptions of the cryptosystems.

EC-NS is constructed in a manner similar to KMOV [14] whereby factoring based algorithms are exported to particular families of elliptic curves². The applicable curves have the specific form:

$$E_n(0, b) : y^2 = x^3 + b \pmod{n}, \text{ for } b \in \mathbb{Z}_n^*$$

with $p \equiv q \equiv 2 \pmod{3}$ and $\mu = |E_n(0, b)| = lcm(p+1, q+1)$. Further requirements are as follows:

$$p+1 = 6 \times u \times p', \text{ where } u = \prod p_i^{\delta_i}$$

$$q+1 = 6 \times u \times q', \text{ where } u = \prod q_i^{\delta_j}$$

for \mathcal{B} -smooth integers³ u and v of (roughly) equal bit size such that $gcd(6, u, v, p', q') = 1$, primes p', q' and $B = O(\log n)$. By the properties of primes p and q , the two curves $E_p(0, b)$ and $E_q(0, b)$ are cyclic groups of orders $p+1$ and $q+1$, respectively.

Let G be a (base) point of $E_n(0, b)$ such that its order is a multiple of $\sigma = uv$. Then encryption of a plaintext $m \in \mathbb{Z}_\sigma$ can be realized as

$$Enc(m) = C = (m + \sigma r)G, \text{ where } r \in_R E_n(0, b)$$

Because σ is \mathcal{B} -smooth, it possible to efficiently compute discrete logarithms for a base of degree σ by using a combination of the baby-step giant-step and Pohlig-Hellman algorithms [15]. Thus, with the knowledge of μ , decryption can be accomplished by computing the discrete logarithm of $(\mu/\sigma)C$ with respect to the base $G' = (\mu/\sigma)G$. The security of the scheme is equivalent to computing residue classes on $E_n(0, b)$. The following outlines the cryptosystem:

EC-OU uses the fact that discrete logarithms are easy to compute in curves $E_p(\overline{a_p}, \overline{b_p})$ over F_p which have trace of

²The KMOV paper introduced elliptic curve schemes that, like the RSA cryptosystem, base their security of the difficulty of factoring a value $n = pq$, where p, q are large primes. This differs from typical ECC solutions that base themselves on the computationally hard discrete logarithm problem.

³An integer is said to be \mathcal{B} -smooth if all its prime factors are $\leq \mathcal{B}$.

Elliptic Curve Naccache-Stern (EC-NS) Encryption

Public Key	$n = pq, b, \sigma, G, E_n(0, b)$
Private Key	(p, q) or $\mu = lcm(p+1, q+1)$
Encryption	plaintext $m \in \mathbb{Z}_\sigma$, $r \in_R \mathbb{Z}_n$, ciphertext $C = (m + \sigma r)G$
Decryption	compute $u = (\mu/\sigma)C = mG'$. Use Pohlig-Hellman & baby-step giant-step to compute the discrete log of u in base G'

Frobenius one (anomalous curves)⁴, where values $\overline{a_p}, \overline{b_p}$ denote a particular curve. Paillier extends this discrete logarithm recover ability property to a p -subgroup of $E_{p^2}(a, b)$ such that the projection onto F_p gives the twist of an anomalous curve.

Define $n = p^2q$, where p, q are large 341-bit primes, and $p \equiv 2 \pmod{3}$. Values $\overline{a_p}, \overline{b_p} \in F_p$ are chosen such that $E_p(\overline{a_p}, \overline{b_p})$ is of order $p+2$. A random curve $E_q(\overline{a_q}, \overline{b_q})$ along with a lift $E_{p^2}(a_p, b_p)$ of $E_p(\overline{a_p}, \overline{b_p})$ to F_{p^2} are chosen. Then, by using the Chinese Remainder Theorem (CRT), $E_{p^2}(a_p, b_p)$ and $E_q(\overline{a_q}, \overline{b_q})$ are combined to get the curve $E_n = E_n(a, b)$, where $a, b \in \mathbb{Z}_n$. A base point $G \in E_n$ of maximal order $lcm(|E_{p^2}|, |E_q|)$ is chosen and $H = nG$. The cryptosystem's security can be shown equivalent to factoring $n = p^2q$.

Elliptic Curve Okamoto-Uchiyama (EC-OU) Encryption

Public Key	$n = p^2q, G, H, E_n$
Private Key	p
Encryption	plaintext $m < 2^{k-1}$, $r \in_R 2^{2k}$, ciphertext $C = mG + rH$
Decryption	compute $m = \frac{\psi_p((p+2)C)}{\psi_p((p+2)G)} \pmod{p}$

where $\psi_p(x, y) = -\frac{x}{y} \pmod{p^2}$ and has the morphic property that if $P = mG$ for arbitrary points P, G , then

$$m = \frac{\psi_p(P)}{\psi_p(G)} \pmod{p}$$

provided that $G \neq \mathcal{O}_{p^2}$.

The cryptoscheme Elliptic Curve Paillier (EC-P) extends the settings of EC-OU to curves defined over \mathbb{Z}_{n^2} , where $n = pq$ and p, q are large primes with the properties that $p \equiv q \equiv 2 \pmod{3}$. Values $\overline{a_p}, \overline{b_p} \in F_p$ and $\overline{a_q}, \overline{b_q} \in F_q$ are chosen such that $E_p(\overline{a_p}, \overline{b_p})$ is of order $p+2$ and $E_q(\overline{a_q}, \overline{b_q})$ is of order $q+2$. The lifted curves $E_{p^2}(\overline{a_p}, \overline{b_p})$ and $E_{q^2}(\overline{a_q}, \overline{b_q})$ are chosen and combined to get $E_{n^2}(a, b)$. A base point $G \in E_{n^2}$ of order divisible by n is chosen, possibly of maximal order $n\mu$, where $\mu = \mu(n) = lcm(p+2, q+2)$. The security of EC-P is based upon the problem of computing residuosity classes over E_{n^2} . Here is the scheme:

It is important to note that in [8], S. Galbraith shows that the use of anomalous curves in the way it is described in the two schemes above is insecure. The attack reveals the private key by efficiently extracting it from the public key. Although the same author proposes a variation of the EC-P scheme, this new scheme is not as efficient and therefore requires too much computation for the scenarios we are considering.

⁴Specifically, such a computation of discrete logarithms requires $O(\log^3 p)$ bit operations.

Elliptic Curve Paillier (EC-P) Encryption

Public Key	$n = pq, G, E_{n^2}$
Private Key	$\mu = lcm(p+2, q+2)$ or equivalently (p, q)
Encryption	plaintext $m \in \mathbb{Z}_m$, $r \in_R \mathbb{Z}_n$, ciphertext $C = (m + nr)$
Decryption	compute $m = \frac{\psi_n(\mu C)}{\psi_n(\mu G)} \pmod n$

We now describe the elliptic curve ElGamal encryption scheme (EC-EG). This is equivalent to the original ElGamal scheme [7] but transformed to an additive group. Key set-up consists in choosing an elliptic curve E together with a 163-bit prime p and generator G . Its security is based upon the Elliptic Curve Discrete Log Problem (ECDLP).

ElGamal Encryption Scheme (EC-EG)

Public Key	$E, p, G, Y = xG$, where $G, Y \in F_p$
Private Key	$x \in F_p$
Encryption	plaintext $M = map(m)$, $r \in_R F_p$, ciphertext $C = (R, S)$, where $R = kG, S = M + kY$
Decryption	$M = -xR + S = -xkG + M + xkG$, $m = rmap(M)$

EC-EG is additively homomorphic (see section IV) and ciphertexts are combined through addition. The summation of two EC-EG ciphertexts requires two point additions, namely one for each of the ciphertext components R and S .

$map()$ refers to the mapping function used to map values (e.g. plaintexts) into points on the curve, and vice versa. Such a function is necessary because the operands of elliptic curve operations are elliptic curve points. This mapping needs to be deterministic such that the same plaintext always maps to the same point. Note that the operation used to map a value is independent from the transformation used to encrypt it: encryption simply transforms a point into another point on the elliptic curve. There exist standard mapping functions but we require one that has the additional property of being homomorphic, i.e. $map(m_1 + m_2) = map(m_1) + map(m_2)$, as suggested in [13].

VI. ANALYSIS AND RESULTS

We compare the performance of a selected sub-set of the public-key cryptosystems described in the previous section. We choose those schemes that meet a large portion of the desirable criteria set forth in Section IV. These include the OU, Benaloh and four ECC cryptosystems. All the selected schemes are probabilistic cryptosystems and meet our security requirement of being *provably computationally secure*. The following are the specific aspects of the schemes that we wish to compare:

- **encryption at sensors:** The encryption transformation needs to be relatively cheap computationally to minimize the energy.

- **homomorphic operation:** takes place at aggregator nodes and is computed less frequently than encryption.

- **bandwidth:** the size of a ciphertext, which is typically larger than the plaintext, affects the number of bits/packets required to transmit the value.

- **decryption:** performing decryption is the responsibility of the reader/querier, which is assumed to be a powerful machine, and may be an offline operation.

For WSNs, we perceive (1) and (2) to be the most important factors, as they will be the operations performed most frequently at sensors. The combination of ciphertexts with the use of homomorphic operations would seemingly happen only at aggregators and possibly intermediary forwarding nodes. Decryption is the least important criterion because it is typically done at a powerful device. We now focus our attention on the candidates that we deemed interesting in section V.

A. Common Unit of Measurement

Since we are comparing schemes built upon different mathematical foundations (computations over elliptic curves versus finite fields), it becomes important to have a *base unit* of measurement common to all schemes in order to fairly compare them. We choose that unit to be *1024-bit modular multiplications* and follow the same methodology for comparison as in [10]. To convert the addition of a point on an elliptic curve over F_p to base units, we first measure the number of $|p|$ -bit modular multiplications required, and then convert these to 1024-bit modular multiplications.

The computation of our focus is xG over F_p , where x is a $|p|$ -bit scalar and G is a point on the curve. Similarly to the square-and-multiply method in finite fields (used during modular exponentiations), we apply the double-and-add algorithm, requiring $|p|$ doublings and $1/2|p|$ additions. Each point doubling/addition operation involves the computation of an inverse which is approximately equivalent to 3 multiplications. With the additional 2 multiplications that take place, we count 5 total multiplications per point addition and doubling, and therefore a total of $5 \times \frac{3}{2} \times |p| = \frac{15}{2} \times |p|$ modular multiplications to compute xG . Next, we need a method for comparing the cost of modular multiplications over different sized moduli. Note that a y -bit modular multiplication has a complexity of $O(y^2)$. One can then conclude that a y -bit modular multiplication is approximately equivalent to $\frac{y^2}{1024^2}$ 1024-bit modular multiplications.

As an example, the computation xG over F_p , where p is a 163-bit prime, requires on average $245 = 163 + 82$ point doublings and additions, i.e. $1225(245 * 5)$ 160-bit modular multiplications. A 1024-bit modular multiplication is approximately 40 times more expensive than that of a 163-bit one, and so, computing xG requires approximately $1225/40 = 31$ 1024-bit modular multiplications.

B. Results

Table I compares the performance of the public key homomorphic encryption candidates described in section V in terms of their computational costs. All table entries consist of two values: (1) the formulas used to determine the respective costs

TABLE I

PERFORMANCE COMPARISON OF CANDIDATES: (1) FORMULAS AND (2) NUMBER OF COMPUTATIONS (1024-BIT MODULAR MULTIPLICATIONS) AND BANDWIDTH (NUMBER OF BITS).

Scheme	Encryption		Addition		Decryption		Bandwidth	
EC-NS	$\frac{15}{2} n $	1800	$5 n $	5	$\frac{15}{2} n $	7680	$ n + 2$	1026
EC-OU	$5 + \frac{15}{2} m + \frac{15}{2} 2k $	690	$5 n $	5	$\frac{15}{2} p $	2558	$ n + 2$	1026
EC-P	$\frac{15}{2} n^2 $	33105	5	20	14	56	$ n^2 + 4$	2052
EC-EG	$2\frac{15}{2} p + \frac{15}{2} m $	38	$10 p $	1	$\frac{15}{2} p + 5 + \text{map}$	97	$2(p + 1)$	328
OU	$\frac{3}{2}(m + r) + 1$	132	$1 n $	1	$\frac{3}{2} p $	512	$ n $	1024
Benaloh	$\frac{3}{2}(m + r) + 1$	(12+r(max))	$1 n $	1	\sqrt{r}	505	$ n $	1024

and (2) the actual number of computations and bits transmitted when applying the formulas to our set of assumed values, as described below. The formulas refer to parameters of the respective schemes, i.e. the p in EC-EG refers to the 163-bit modulus defining the elliptic curve, while the p in EC-OU is the (typically) 341-bit prime that is used to construct the modulus n . All computations (the second part of the entries) are converted to and measured in terms the number of *base units*, which we previously defined to be 1024-bit modular multiplications. Note that the formulas for EC-EG and EC-P reflect the number of 163-bit and 2048-bit modular multiplications, respectively⁵.

Parameters have been selected such as to obtain an equal 1024-bit security level amongst all schemes and to reflect an envisioned WSN setting. For *EC-NS*, *EC-OU*, *EC-P* and *OU*, primes p and q are selected such that $|n| = 1024$, while we use one of the standard (IEEE) ECC curves over F_{163} defined in [11]. Random nonces are assumed to be 80-bits while plaintexts m are 8-bit values. As for the decryption with EC-EG and Benaloh, we need to specify an upper limit on the number values being aggregated, and we set this to 1000⁶.

One inherent difference between many of the schemes is the size of the operands, as EC-EG uses significantly smaller values than the other considered schemes (163-bit moduli versus ≥ 1024 -bits and more), affecting both the performance and bandwidth. EC-P shows itself to be especially costly, mainly due to its use of a 2048-bit modulus. Each 2048-bit modular multiplication is equivalent to four 1024-bit ones. With no particular advantage standing out, we can eliminate it as an attractive final candidate scheme. To represent a point on an elliptic curve one needs to encode a pair of values, namely the x and y coordinates, both of size equal to the modulus. Using the point compression technique, one only needs the x coordinate together with one or more extra bits of the y coordinate⁷ in order to represent the point. EC-EG benefits

⁵As modular multiplications with 2048-bit moduli are approximately 4 times more expensive than with 1024-bit moduli, we convert the 14 2048-bit modular multiplications in the decryption in EC-P to 56 1024-bit modular multiplications.

⁶If 1000 8-bit values can be aggregated, then the maximum possible aggregate value is 255,000 and for the Benaloh encryption scheme, r is set to this.

⁷When working over F_p , where p is prime, only one extra bit is necessary since these exist only two solutions to the Weierstrass equation for a given x , namely the quadratic residues. However, for curves over \mathbb{Z}_n , where $n = pq$, there exist 4 solutions, and two bits are therefore required.

from its smaller modulus and achieves the smallest ciphertext of any scheme.

Of particular interest are the results of Gura et al. [10] in which the authors implement optimized algorithms for modular multiplications on the ATmega128 8-bit processors, which are the same as those used by the popular Mica2 sensor nodes from Crossbow [1]. With these algorithms, they measure the elliptic curve computation xG , of which there are two instances of in EC-EG encryption, to complete in approximately 0.81 seconds. In light of these results we view certain public-key schemes to be practical when implemented over sensor nodes.

Three of the six candidate schemes stand out above the others in their superior performance, namely *OU*, *Benaloh* and *EC-EG*. Common to all three is that encryption performance depends upon the size of the plaintext (m), which is an advantage when using our assumed parameters (small plaintexts), which indeed are realistic for WSNs.

C. Limitations of EC-EG and Benaloh

Parts of the decryption functions for the two cryptosystems EC-EG and Benaloh are both based on brute-force techniques, and this limits their usability in certain WSN settings when the brute-force space becomes too large. With EC-EG it is necessary to reverse the mapping function such as to map the elliptic curve point (M) back to a numeric value (m) through repeated point additions (with the generator G), and with Benaloh one needs to compute exponentiations until $y^{-m'}c \pmod{n} \in Enc(0)$. Both schemes employ a type of big-step little-step algorithm that balance storage and computation such as to speed up their brute-force techniques. They involve storing pre-computed values at selected intervals and then enumerating through the computations until one of these pre-computed values are found.

One typical configuration for big-step little-step algorithms is to store \sqrt{max} pre-computed values at intervals of \sqrt{max} apart, which will then requiring no more than \sqrt{max} computations, where max is the largest possible value that may be represented. For the parameters we chose in section VI-B, with 8-bit plaintexts and at most 1000 values aggregated, we have $max = 2^{18}$. We now look at what effect it would have if sensors were to perform the decryption and we focus on EC-EG, although the same arguments also apply to the Benaloh cryptoscheme. As mentioned earlier, we need 164

TABLE II
SUMMARY OF KNOWN AGGREGATION FUNCTIONS USING ADDITION.

Function	Sensor ($S_i \in \mathcal{S}$)	Aggregator ($A_i \in \mathcal{A}$)	Sink (R)
Average	$Enc(s_i)$	$\sum_{i=1}^n Enc(s_i) n$	$\frac{Dec(\sum_{i=1}^n Enc(s_i))}{n}$
Variance	$Enc(s_i) Enc(s_i^2)$	$\sum_{i=1}^n Enc(s_i) \sum_{i=1}^n Enc(s_i^2) n$	$\frac{Dec(\sum_{i=1}^n Enc(s_i^2))}{n} - (Average)^2$
Mov. Detect.	$B_i = \{b_n = 0, \dots, b_i = s_i, \dots, b_0 = 0\}$ $Enc(B_i)$	$\sum_{i=1}^n B_i$	$Dec(\sum_{i=1}^n B_i) = \{s_n, \dots, s_0\}$
Checksum	$csum_i = \{1 - \sum_{j=0}^{len(s_i)} s_{i,j}\}_{BASE2}$ $Enc(csum_i)$	$\sum_{i=1}^n Enc(csum_i)$	$csum = Dec(\sum_{i=1}^n Enc(csum_i))$ $s = \sum_{i=0}^n s_i$ $csum = \{1 - \sum_{j=0}^{len(x)} s_j\}_{BASE2}$

bits to represent an elliptic curve point on a curve defined by a 163 bit prime p . This means that the storage required by the big-step little-step algorithm would be $2^9 \times 164 \approx 2^{17}$ bits, or 16 Kbytes. Although larger Flash memories may soon be available for sensors, we can easily imagine how larger parameters, be it the size of plaintexts or number of aggregated values, can easily exceed the storage capacities of nodes. The same calculations apply to the cost of performing \sqrt{max} computations, as it may take too long to compute or simply drain the power supply once max increases.

VII. APPLICATIONS

A. Data Aggregation

The problem of end-to-end encrypted data aggregation has been addressed in [9] and [3]. Two distinct cases are described: the usage of additive encryption for calculating the average and for movement detection. [3] present an approach to calculate average and variance.

The basic idea in both average and variance calculations is that division is shifted to the sink (R) and performed over the plaintext value after decryption. This way, sensors ($S_i \in \mathcal{S}$) along the path send the value (s_i), encrypted, or the encrypted square of the value in case of the variance, which is added at every aggregator ($A_i \in \mathcal{A}$) to all the encrypted values read by the other sensors, or forwarded by a forwarding node ($F_i \in \mathcal{F}$) in the direction of R . Once the aggregated values reaches the sink, together with the number of participating sensors, it performs decryption and calculates the average or variance from the sum. The **checksum** is a particular case of the **sum** function.

Table II is a representation of the generic aggregation method described above, for different applications. All of these are shown to work in a hierarchical topology of a WSN. While the **sum**, **average**, **variance** and **checksum** aggregation functions are rather intuitive, **movement detection** may seem less straight forward. Every sensor will send either a 1 or a 0 in the bit array position that corresponds to the sensor's relative position to the other nodes in the case it either senses the presence of an object (send a 1) or not (send a 0), respectively. The **sum** of all these values will correspond to a snapshot of the presence of objects in the whole network.

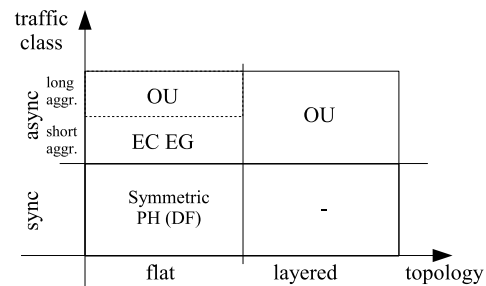


Fig. 2. Recommendations on the usage of additive homomorphic schemes in WSNs.

B. Long-term data storage

One other application of data aggregation is related to long-term data storage. In contrast to in-network processing while information flows in the network, this application relies on the fact that data is kept in the nodes for later retrieval [20]. Since the nodes have restricted storage capacity, it is important to reduce the amount of values that are actually stored. Depending on the application, these policies may vary depending on time, amount of data and on the actual maximum storage space available.

We can therefore apply the same concepts of data aggregation, as we did for in-network processing, to reduce the amount of data stored at the nodes by condensing and accumulating the information that is to be saved over the time.

VIII. RECOMMENDATION

In a *synchronous sensor network* the values are fluctuating and adding security should not impact the reactive and real-time responsiveness of the system. For this reason, and due to the very restricted lifetime of the values, the authors still support the conclusions of the work presented in [9] and [3] on the application of symmetric privacy homomorphisms with all its security weaknesses but performance benefits. The matter of applying a similar scheme to *layered* topologies which require in-network decryption is one which we believe is not solvable by these approaches.

As to *asynchronous sensor networks*, we believe we have to consider the problem when applied to a *flat* and to a *layered* topology separately. *Layered* topologies, as described

in Section III, require intermediate nodes to be able to decrypt values. In Section VI, limitations on decryption, related to the value space and algorithm, are presented for EC-EG and Benaloh which make them unsuitable for implementation on the sensor nodes. We consider that in this case, the best candidate is OU since it provides the best ratio between encryption and decryption costs. This comes at the cost of a bigger ciphertext size, which we still consider acceptable for applications which require only seldom polling and aggregation of the values.

In a *flat* distribution of the network, we have to consider the threshold at which the constant addition of the values, or the initial size of the sensed values, affects the feasibility of decryption, as stated in Section VI. The primary candidate, both in computation effort and bandwidth, is EC-EG. However, this scheme suffers from an expensive mapping function during decryption which, in some cases, may become too costly to revert. Once this threshold is reached, we believe that OU is another possible candidate. The ciphertext size pushes it to a second, still viable, solution, where applying EC-EG is no longer possible. Figure 2 provides a summary.

Furthermore, the application has a direct impact on the scheme to use. Calculating the **minimum** and **maximum**, as stated in [9], is not possible to achieve due to an inherent problem of using privacy homomorphisms. For such applications we propose the usage of a scheme proposed in [2] which makes use of an Order Preserving Encryption Scheme [18] and applies it to a sensor network scenario. When calculating the **variance**, the problem of the value space appears once again: since the aggregated data is actually the square of the sensed value, the value space doubles and may easily reach values no longer feasible for applying EC-EG. This case would be another possibility where to apply OU.

There is also a contradicting argument when applying EC-EG or OU for long-term storage. On one hand, the number of aggregated values is quite high, making the reverse mapping function for EC-EG costly, while on the other hand, the ciphertext size of OU is rather large which conflicts with the objective of minimizing the storage space necessary.

IX. CONCLUSION

In this paper we looked at several additive homomorphic public-key encryption schemes and their applicability to WSNs when implemented on computationally limited sensor devices. These cryptosystems permit the addition of ciphertexts which enables secure in-network processing of data such as to minimize bandwidth overhead and thereby reduce the sensors' energy consumption, even in the face of compromised nodes.

We compared the public-key schemes in terms of their computational costs for the most commonly executed cryptographic operations, namely encryption, addition of ciphertexts and decryption, and also the size of their ciphertexts which affects the bandwidth usage at nodes. From these results, we came to view the three encryption algorithms EC-EG, Benaloh and OU, as superior to the other candidate schemes reviewed.

We then listed a few aggregation functions which can be computed over enciphered data and recommended which cryptosystem should be used in which application. We concluded that public-key cryptoschemes are required for specific WSN settings where symmetric encryption schemes prove to be insufficient due to their sensitivity to node compromise.

tosystem should be used in which application. We concluded that public-key cryptoschemes are required for specific WSN settings where symmetric encryption schemes prove to be insufficient due to their sensitivity to node compromise.

X. ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their comments and insights on our work and for bringing to our attention relevant documents.

Results of this work are in preparation for UbiSec&Sens⁸.

REFERENCES

- [1] Crossbow Technology Inc. <http://www.xbow.com/>.
- [2] M. Acharya, J. Girao, and D. Westhoff. Secure comparison of encrypted data in wireless sensor networks. *WiOpt2005*, 2005.
- [3] C. Castelluccia and E. Mykletun and G. Tsudik. Efficient Aggregation of encrypted data in Wireless Sensor Networks. *Mobile and Ubiquitous Systems: Networking and Services*, 2005.
- [4] D. Naccache and J. Stern. A New Public Key Cryptosystem Based on Higher Residues. *ACM Conference on Computer and Communications Security*, pages 59–66, 1998.
- [5] D. Dolev and A.C. Yao. On the security of Public-Key Protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.
- [6] J. Domingo-Ferrer. A Provably Secure Additive and Multiplicative Privacy Homomorphism. *Information Security Conference*, pages 471–483, 2002.
- [7] T. ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *CRYPTO*, IT-31(4):469–472, 1985.
- [8] S. Galbraith. Elliptic Curve Paillier Schemes. *Journal of Cryptology*, 15:129–138, 2002.
- [9] J. Girao, D. Westhoff, and M. Schneider. Cda: Concealed data aggregation for reverse multicast traffic in wireless sensor networks. In *IEEE International Conference on Communications (ICC2005)*, Seoul, Korea, May 2005.
- [10] N. Gura, A. Patel, A. Wander, H. Eberle, and S.C. Shantz. Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. *Cryptographic Hardware and Embedded Systems (CHES)*, pages 119–132, 2004.
- [11] IEEE. Standard P1363: Standard Specifications For Public-Key Cryptography. <http://grouper.ieee.org/groups/1363/>.
- [12] J. Benaloh. Dense Probabilistic Encryption. *Proceedings of the Workshop on Selected Areas of Cryptography*, pages 120–128, 1994.
- [13] J.M. Adler and W. Dai and R. L. Green and C.A. Neff. Computational Details of the VoteHere Homomorphic Election System. *ASIACRYPT*, 2000.
- [14] K. Koyama and U. M Maurer and T. Okamoto and S.A. Vanstone. New Public-Key Schemes Based on Elliptic Curves over the Ring Z_n . *CRYPTO*, pages 252–266, 1991.
- [15] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
- [16] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. *EUROCRYPT*, pages 223–238, 1999.
- [17] P. Paillier. Trapdooring Discrete Logarithms on Elliptic Curves over Rings. *ASIACRYPT*, pages 573–584, 2000.
- [18] R. Agrawal and J. Kiernan and R. Srikant and Y. Xu. Order Preserving Encryption for Numeric Data. *ACM SIGMOD*, 2004.
- [19] T. Okamoto and S. Uchiyama. A New Public-key Cryptosystem as Secure as Factoring. *EUROCRYPT*, pages 308–318, 1998.
- [20] D. Westhoff, J. Girao, and E. Mykletun. TinyPEDS: Tiny Persistent Encrypted Data Storage in Asynchronous Wireless Sensor Networks. *In Submission*, 2005.
- [21] W.R. Heinzelman and A. Chandrakasan and H. Balakrishnan. Energy-efficient Communication Protocols for Wireless Microsensor Networks. *Hawaiian Int'l Conference on Systems Science*, 2000.

⁸The work presented in this paper was supported in part by the European Commission within the STREP UbiSec&Sens of the EU Framework Program 6 for Research and Development (IST-2004-2.4.3). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the UbiSec&Sens project or the European Commission.