

Secure Comparison of Encrypted Data in Wireless Sensor Networks

Mithun Acharya

Department of Computer Science
North Carolina State University
Raleigh North Carolina USA 27695
mpachary@csc.ncsu.edu

Joao Girao

NEC Europe Ltd. Network Laboratories
Heidelberg Germany 69115
joao.girao@netlab.nec.de

Dirk Westhoff

NEC Europe Ltd. Network Laboratories
Heidelberg Germany 69115
dirk.westhoff@netlab.nec.de

Abstract

*End-to-end encryption schemes that support operations over ciphertext are of utmost importance for commercial private party Wireless Sensor Network implementations to become meaningful and profitable. For Wireless Sensor Networks, we demonstrated in our previous work that Privacy Homomorphisms, when used for this purpose, offer two striking advantages apart from end-to-end concealment of data and ability to operate on ciphertexts: flexibility by keyless aggregation and conservation and balancing of aggregator backbone energy. We offered proof of concept by applying a certain Privacy Homomorphism for sensor network applications that rely on the addition operation. But a large class of aggregator functions like median computation or finding maximum/minimum rely exclusively on comparison operations. Unfortunately, as shown by Rivest, et. al., any Privacy Homomorphism is insecure even against ciphertext only attacks, if they support comparison operations. In this paper we show that a particular order preserving encryption scheme achieves the above mentioned energy benefits and flexibility when used to support comparison operations over encrypted texts for Wireless Sensor Networks, while also managing to hide the plaintext distribution and being secure against ciphertext only attacks. The scheme is shown to have reasonable memory and computation overhead when applied for Wireless Sensor Networks.*¹

¹The work presented in this paper was supported in part by the European Commission within the integrated Project DAIDALOS of the EU Framework Programme 6 for Research and Development (IST-2002-506997)(<http://www.ist.daidalos.org>).

1 Introduction

Wireless Sensor Networks have received considerable academia research attention and are now all set for a widespread commercial implementation for applications varying from wild life tracking systems to health monitoring and environmental surveillance applications. We envision a future wherein different private parties develop proprietary sensor network implementations for the same task and that would propel a competition trend in sensor network industry reminiscent of the mobile phone business model. It is also very likely that private party wireless sensor networks with common interest would form network of networks to collaborate in some meaningful way. In this scenario, encryption of sensed data attains primary importance, without which it would be meaningless to implement commercial private party Wireless Sensor Network solutions for various tasks.

Efficient utilization of energy becomes another concern in power-constrained Wireless Sensor Networks. A typical application scenario for a WSN is to sense the data of interest and to transmit them to a central point called sink. The sensed data needs to be analyzed, which eventually serves to initiate some action. Analysis in most scenarios presumes computation of an optimum, e.g., the minimum or maximum, the computation of the average, or the detection of some movement pattern. This analysis can be either done at the sink node if all the sensors directly submit the sensed values to the sink or by the network itself, in a hierarchical fashion. The latter is beneficial in order to reduce the amount of data to be transmitted over wireless connections. To save the overall energy resources of the network, it is agreed that the sensed data needs to be consolidated and aggregated on the way to its final destination. The nodes that

perform the aggregation operation are known as aggregators.

In [7, 8], some of the authors introduced the concept of Concealed Data Aggregation (CDA), wherein we addressed the security and energy requirements for Wireless Sensor Networks mentioned above. CDA considers a special class of encryption functions, called *Privacy Homomorphisms*, that allow end-to-end encryption between the sensors and the sink node and simultaneously provide aggregators with the possibility to carry out aggregation functions that are applied to ciphertexts. Privacy Homomorphism is an encryption transformation that allows direct computation on encrypted data. For plaintext operands a and b and key k , we have $a + b = D_k(E_k(a) + E_k(b))$, if E_k and D_k are additively homomorphic encryption and decryption transformations. This provides the advantage that intermediate aggregators do not have to carry out decryption and encryption operations and thus do not need to store the sensitive data (key or the decrypted data). To our best knowledge, CDA is the first work focusing on end-to-end encryption in Wireless Sensor Networks by still providing in-network processing. CDA uses Domingo-Ferrer [5] Privacy Homomorphism to support applications which rely on addition and multiplication aggregation operations. Domingo-Ferrer Privacy Homomorphism is additively and multiplicatively homomorphic and secure against chosen ciphertext attacks. In [15], Rivest et. al., show that a Privacy Homomorphism is insecure even against ciphertext only attack if it supports comparison operations. In this work, we show how CDA can be used in applications that rely exclusively on comparison operations, while being immune to ciphertext only attacks.

2 Motivation and Related Work

As aforementioned, apart from concealment by encryption, it should also be possible for sensor nodes to route and operate on ciphertexts to make any collaboration trustworthy and mutually beneficial for the involved parties. Hop-by-hop encryption schemes [11] that use RC5 provide concealment but do not support ciphertext operations since the aggregators will have to decrypt the data before operating on them. Motivated by this, we demonstrated in [8] that Privacy Homomorphisms, when used for data aggregation, offer two striking advantages in the form of flexibility by keyless aggregation and conservation and balancing of aggregator backbone energy. Load balancing can be achieved by evenly distributing the energy consuming task of aggregation among all the keyless aggregators, since the operations can be performed directly on the ciphertexts. The aggregators need not decrypt and re-encrypt each sensed data on its way to the sink which saves a lot of aggregator backbone energy. Also, a keyless aggregator compromised by an

adversary neither gives out the aggregated value nor the key. We provided a proof of concept in [7, 8], wherein we applied the Domingo-Ferrer Privacy Homomorphism [5] for Concealed Data Aggregation to support movement detection and average computation applications. However, the ciphertext operation was limited to the addition operation and average computation.

A large class of useful sensor network applications such as median computation or finding maximum/minimum, rely exclusively on comparison operations. Boundary values (maximum and minimum) values are crucial and need to be immediately reported in measurement-critical applications such as fire monitoring systems, temperature sensors, pressure sensors and radiation level sensors. [15] points out that any Privacy Homomorphism which supports comparison operation is insecure even against ciphertext only attacks if the domain of sensed values is known. Recently an order preserving encryption scheme, called OPES [2], was proposed by Agrawal, et al. for database encryption. This scheme is secure against ciphertext only attacks if the domain is assumed to be secret and its main objective is to hide the distribution of plaintext values. It is important to hide the plaintext distribution since an attacker can launch a tight estimation attack with the knowledge of distribution and glean a lot of information. In this paper, we apply OPES for CDA, where the aggregation operation is a comparison based function such as *maximum* or *minimum*. We show that OPES achieves the energy benefits and flexibility we demonstrated in [8] when used to support secure comparison operations over the encrypted values for Wireless Sensor Networks, apart from hiding the distribution of sensed values and being secure against ciphertext only attacks. We also show that the memory requirements for implementing OPES for Wireless Sensor Networks are reasonable on the MICA2 Motes [1], our reference platform. Furthermore, both, the comparison operation and OPES are applicable in a cascaded manner by several aggregating nodes. This makes the scheme also valid for large Wireless Sensor Networks. To the best of our knowledge, ours is the first work that attempts to support encrypted comparison in Wireless Sensor Networks in an energy and memory efficient way. We note here that other security goals such as integrity and authentication, although they are definitely mandatory, are outside the scope of this paper. We exemplarily refer to [13, 17, 16].

The rest of the paper is organized as follows. In Section 3, we present the network and the threat model and clearly state the problem. In Section 4, we show how OPES can be applied for Wireless Sensor Networks for supporting secure ciphertext comparisons. We note that the numerical and statistical methods used in this paper stem directly from the OPES scheme. The objective of this paper is to highlight the energy gains at the aggregator when OPES is used for se-

cure ciphertext comparisons, comparing it with the hop-by-hop encryption alternative. The resulting message, memory and the computation overhead incurred is shown to be reasonable for MICA2 Motes and we comment on this in the next two sections. We conclude in Section 7.

3 Network Model and Problem Definition

The Wireless Sensor Network considered in this work is static and is represented by a graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$ with $|\mathcal{N}|$ nodes and $|\mathcal{L}|$ links. Each node represents a wireless sensor node and each link represents a bidirectional communication channel over a shared medium. There is one single stated node $R \in \mathcal{N}$, namely the sink node with virtually unlimited power and storage capacity ideally placed in the centre of the plane covered by \mathcal{G} . Per epoch t during the lifetime of the Wireless Sensor Network a set of aggregator nodes $\mathcal{A}_t \subset \mathcal{N}$, a set of forwarding nodes $\mathcal{F}_t \subset \mathcal{N}$, a set of sensing nodes $\mathcal{S}_t \subset \mathcal{N}$ and a set of idle nodes $\mathcal{I}_t \subset \mathcal{N}$ with $\mathcal{A}_t \cap \mathcal{F}_t \cap \mathcal{S}_t \cap \mathcal{I}_t = \emptyset$ and $\mathcal{A}_t \cup \mathcal{F}_t \cup \mathcal{S}_t \cup \mathcal{I}_t = \mathcal{N}$ is elected. Let x_{in} and x_{out} be the number of ingoing and outgoing messages at a node x for reverse multicast traffic to the R . If $(x_{in}, x_{out}) = (0, 1)$ then $x \in \mathcal{S}_t$, if $(x_{in}, x_{out}) = (1, 1)$ then $x \in \mathcal{F}_t$, and finally if $(x_{in}, x_{out}) = (n, 1)$, $n > 1$ then $x \in \mathcal{A}_t$. For R it holds $(x_{in}, x_{out}) = (n, 0)$. All $x \in \mathcal{S}_t$, the sensing nodes, perform $c = OPES(p)$, where c is the encryption of sensed plaintext value p , on using the *OPES* scheme. We assume that $p \in [p_{min}, p_{max}]$, where $p_{min} \geq P_{min}$ and $p_{max} \leq P_{max}$. $[P_{min}, P_{max}]$ defines the domain for the sensed values. All sensor nodes are assumed to follow the same distribution over the sensed values. Before the deployment of the sensors, the network designer will have access to a sample of $|P|$ sensed values, $p_1 < p_2 \dots < p_{|P|}$, which will be used to approximate the distribution during sensor predeployment stage. We call this as the *input distribution*, \mathcal{P} . All $x \in \mathcal{A}_t$, the aggregating nodes, compute the aggregation function $c_{opt} = opt(c_1, c_2, \dots, c_n)$ on the received encrypted data c_i , $i = 1, 2, \dots, n$ with $opt : \{0, 1\}^k \times \dots \times \{0, 1\}^k \rightarrow \{0, 1\}^k$. opt could be either *min* or *max* operation. All $x \in \mathcal{F}_t$, the forwarding nodes just forward the incoming data. All $x \in \mathcal{I}_t$ are not available in epoch t . At epoch $t + 1$, \mathcal{A}_{t+1} varies from \mathcal{A}_t where $|D|$ with $D = \mathcal{A}_t \cap \mathcal{A}_{t+1}$ is a metric for the value of the aggregator nodes' election process. Consequently also \mathcal{F}_{t+1} , \mathcal{S}_{t+1} and \mathcal{I}_{t+1} may differ from their counterparts in epoch t . Protocols like LEACH [10] may be used for determining \mathcal{A}_t , \mathcal{F}_t , \mathcal{S}_t and \mathcal{I}_t , for a given epoch t .

For the threat model, we assume that an attacker has access to any number of ciphertext values but he does not have access to any ciphertext-plaintext pairs. The sensors report the sensed values with granularity g . We assume that p_{min} , p_{max} and g are not known to the attacker. We also assume

that target distribution specified by the network designer, \mathcal{T} , to be a secret (details in the next section). The problem now is to compute the comparison functions securely and efficiently at the aggregators over the encrypted texts in such a way that the attacker can neither predict the sensed plaintext values nor the distribution of plaintext values with access to any number of ciphertexts. To achieve a reasonable security level under the restriction of nodes without any tamper-resistant unit, we propose to combine this approach with the novel key pre-distribution scheme *topology aware group keying* [8].

4 Adapting OPES Scheme for Encrypted Comparisons in Wireless Sensor Network

Most of the order preserving encryption schemes like [3], [12] and [9] allow the attacker to guess the nature of the plaintext distribution by graphing the distribution pattern for the observed ciphertexts. This is possible since the ciphertext distribution depends on the plaintext distribution for these encryption schemes. The basic idea of OPES is to take as input a user-provided *target distribution*, which we call \mathcal{T} henceforth, and transform the plaintext values in such a way that the transformation preserves the order while the transformed values follow the target distribution. OPES decreases the comparison query process times for encrypted databases since no decryption and re-encryption is necessary. By shifting most of the computationally intensive OPES operations to the sink node R (where power and computation power is virtually unlimited), during predeployment stage, we show that once the sensor networks are deployed, the memory and computation overhead is reasonable for MICA2 Motes. In this section we show how we adapt OPES to the sensor network scenario.

As given in [2], OPES has the following stages:

1. *Model*: The input distribution \mathcal{P} and the target distributions \mathcal{T} are modeled as piecewise linear splines.
2. *Flatten*: The input plaintext distribution \mathcal{P} is transformed into a "flat" distribution \mathcal{F} such that the values in \mathcal{F} are uniformly distributed.
3. *Transform*: The flat distribution \mathcal{F} is transformed into cipher distribution \mathcal{C} such that the values in \mathcal{C} are according to the network designer specified target distribution \mathcal{T} .

In the **Modeling** phase, the sorted points $p_1 < p_2 \dots < p_{|P|}$ (the sensor sample values available to the network designer) are split into a number of buckets, with each bucket represented by the bucket boundaries $[p_l, p_h)$, where p_l is the least value and p_h the greatest value. For a given bucket $[p_l, p_h)$, with $h - l - 1$ sorted points, $\{p_{l+1}, p_{l+2}, \dots, p_{h-1}\}$,

the distribution within the bucket is approximated by a linear spline. A point $p_l \leq p_s \leq p_h$ is identified which deviates the maximum from this linear spline among all p_i 's, $l + 1 \leq i \leq h - 1$ and the bucket is recursively split here. The splitting is stopped when the number of points in the bucket is below some threshold. Subsequently some buckets are merged into bigger buckets. The idea of merging is to minimize the number of buckets and yet have the values within the buckets after mapping be uniformly distributed. OPES uses Minimum Description Length (MDL) principle [14] to achieve this balance. The whole process happens at the sink during predeployment. The bucket boundaries are uploaded onto each sensor. For m buckets, it is obvious that the sensors will have to store $m + 1$ bucket boundaries.

In the **Flattening** phase, a plaintext bucket B is mapped into a bucket B^f in the flattened space in such a way that the density in the flattened bucket and over all buckets will be uniform. OPES notices that if a distribution over $[0, p_h)$ has the density function $qp + r$, where $p \in [0, p_h)$, then for any constant $z > 0$, the mapping function $M(p)$ will yield a uniform distribution. $M(p)$ is calculated as

$$M(p) = z(qp^2/2r + p)$$

$s = q/2r$ is called the quadratic coefficient and one quadratic coefficient for each bucket is uploaded to all sensors during the predeployment phase. When the sensors are in operation after predeployment, they need not perform the costly operation of $q/2r$ division as they are precomputed by the sink and preloaded onto the sensor nodes before deployment. z is the scale factor, one for each bucket, which ensures that the inter-bucket density is uniform as well. The scale factor which achieves this for each bucket is calculated as

$$z = Kn/(sw^2 + w)$$

w is the width of the bucket and n is the number of points in the bucket. K is the maximum of minimum of the predicted flattened bucket widths. The scale factor computation also ensures *incremental updatability*². To recall, the scale factor computation is done by the sink node R and are preloaded onto the sensor nodes, one for each bucket, before deployment.

The sensors store this information (bucket boundaries, quadratic coefficients and scale factors) in a data structure \mathcal{K}^f , which will be termed as the encryption key that is used to flatten the sensed values once the sensor networks are deployed in the field. We now need to explain how plaintext sensed values are mapped into flattened values when the sensors sense in real time. We have used

²If a scheme does not support *incremental updatability*, then when sensing a new value p , with $p_i < p < p_{i+1}$, all p_j , $j > i$ needs to be re-encrypted. Once the sensors are deployed, this is impossible.

$[p_{min}, p_{max})$ to represent the domain of sensed values in plaintext (which the adversary does not know) and we let $[f_{min}, f_{max})$ to be the domain of flattened sensed values. Note that $f_{max} = f_{min} + \sum_{i=1}^m w_i^f$, where $w_i^f = M_i(w_i)$. w_i is the length of plaintext bucket B_i and w_i^f is the length of corresponding flat bucket. When a sensor senses a plaintext value p , the sensor node performs a binary search over the $m + 1$ bucket boundaries stored as \mathcal{K}^f to determine the bucket for p . After this p is mapped into the flat value f using the equation

$$f = f_{min} + \sum_{j=1}^{i-1} w_j^f + M_i(p - p_{min} - \sum_{j=1}^{i-1} w_j)$$

The inverse mapping of flat value to plaintext is accomplished by

$$p = p_{min} + \sum_{j=1}^{i-1} w_j + M_i^{-1}(f - f_{min} - \sum_{j=1}^{i-1} w_j^f)$$

$$M^{-1}(f) = (-z \pm \sqrt{(z^2 + 4zsf)})/2zs$$

This is energy consuming since it has a division and a square-root operation. Fortunately, all decryption operations (conversion of flat values to plaintext values) are done only by the sink node R , where power is not a concern.

In the transform phase, we want to map the uniform flattened values according to the network designer specified target distribution \mathcal{T} . In other words, we need to flatten the target distribution and align it with the flattened plaintext distribution. During the predistribution phase, the sink models the target distribution and flattens it by a method similar to that used to flatten the input distribution. The modeling of target distribution yields a set of buckets, $\{B_1^t, B_2^t, \dots, B_k^t\}$ and for each bucket, we have a quadratic factor s^t and a scale factor z^t related by

$$z^t = K_t n_t / (s_t (w_t^2) + w_t)$$

K^t is similar to K in the previous case. Let \hat{B}^f be the bucket in the flat space corresponding to the bucket B^t , with length \hat{w}^f (as defined earlier, w^f is the length of B^f). Now to align the flattened plaintext distribution and the flattened target distribution, a scaling factor L is computed and both the target buckets B^t and the flattened target buckets \hat{B}^f are scaled by a factor of L . L is given by

$$L = (\sum_{i=1}^m w_i^f) / (\sum_{i=1}^k \hat{w}_i^f)$$

Hence the length of the cipher bucket B^c corresponding to the target bucket B^t is given by $w_i^c = L w_i^t$ and the length of the scaled flattened target bucket \hat{B}^f is given by $\hat{w}^f = L \hat{w}^f$. Finally the mapping function M^c for mapping values from the bucket B^c to flat bucket \hat{B}^f is defined by quadratic coefficient $s^c = s^t/L$ and scale factor $z_c = z_t$. The bucket boundaries, and for each bucket the quadratic coefficient z_c and the scale factor s_c are stored in the data

structure \mathcal{K}^c . If $[c_{min}, c_{max}]$ is the domain of ciphertexts, then a flat value f from the bucket \hat{B}_i^f can now be mapped into cipher value c using the equation

$$c = c_{min} + \sum_{j=1}^{i-1} w_j^c + (M_i)^{c-1} (f - f_{min} - \sum_{j=1}^{i-1} \bar{w}_j^f)$$

$$M^{c-1}(f) = (-z \pm \sqrt{(z^2 + 4zsf)}) / 2zs$$

The division and the square root operations are the only energy guzzling operations and we show that if we assume a reasonable network density, our model performs better than hop-by-hop encryption. To map the ciphertext to the flat values, we use the equation

$$f = f_{min} + \sum_{j=1}^{i-1} \bar{w}_j^f + M_i^c (c - c_{min} - \sum_{j=1}^{i-1} w_j^c),$$

This computation is done by the sink node.

The OPES scheme is summarized in Figure 1³. In (a), the input distribution is modeled as piecewise linear splines and the buckets used to transform the plaintext distribution into the flattened distribution are computed. In (b), from the target distribution, buckets are computed for the flattened distribution \hat{B}^f . In (c), the buckets of the flattened target distribution are scaled to equal the range of the flattened distribution computed in (a) and the target distribution is scaled proportionately.

The OPES scheme, when adapted for comparing encrypted comparisons in Wireless Sensor Networks is reasonable energywise, as computationally intensive operations are shifted to the sink node and are performed either during predeployment stage or in an offline manner. The sensor nodes perform minimal computation in real time. The computations performed by the sink node during pre-deployment or offline and the sensor nodes in real time are summarised in Figure 2. Energy details and further simplifications are discussed in Section 6

5 Memory and Message Overhead

If we have m buckets, $\{B_1, B_2, \dots, B_m\}$, each sensor node has to store $m + 1$ bucket boundaries, m quadratic coefficients and m scale factors. Thus the memory overhead would be $(3m + 1) \times 32$, assuming 32 bits for each of these values. In [2], which presents OPES, the authors show that even for datasets with 10 million values, the number of buckets required (for both \mathcal{K}^f and \mathcal{K}^c) is not more than 200. Furthermore for uniform distribution, the number of buckets required is shown to be under 10. So even with 200 buckets, the size of the key data structure $\mathcal{K}^f + \mathcal{K}^c$ is not more than $(3 \times 200 + 1) \times 32 = 19232$ bits or around 4.7 KB. We note that this is comparable to the memory size required for key

³This figure has been obtained from the description of the OPES scheme [2].

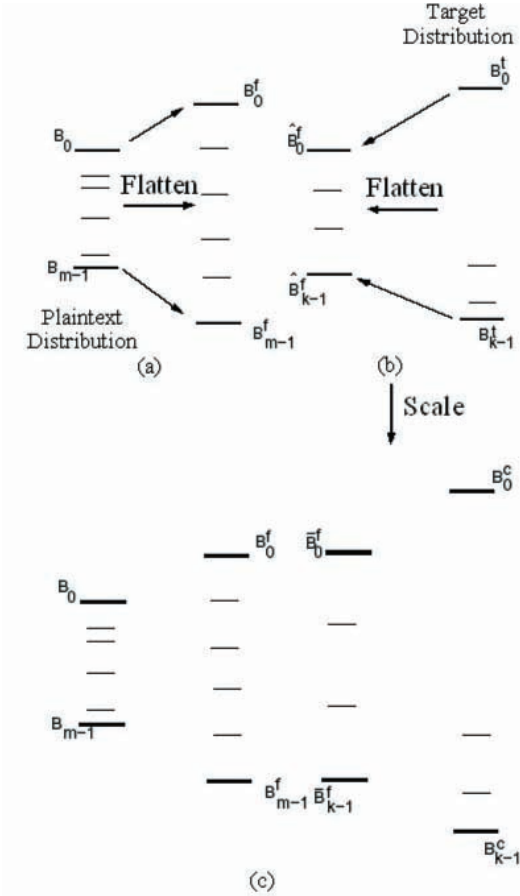


Figure 1. The OPES Scheme

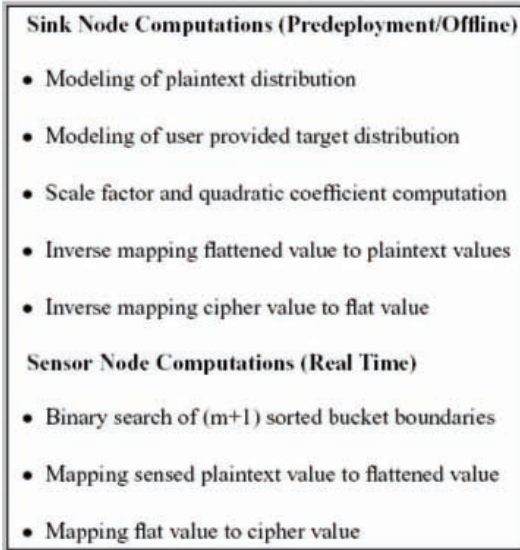


Figure 2. Node Computations

RC5 Encrypt	236
RC5 Decrypt	236
Addition	4
Comparison	4
Multiplication	40
Division	700
Square Root	1500

Table 1. MICA2 Clock Cycle Measurements

ring storage in random key predistribution schemes such as [4] and [6].

Flattening stage ensures that the gap between all the values are uniform. Hence if g_{max} and g_{min} are the maximum and minimum gaps in the plaintext distribution and if most of the gaps are close to g_{max} , then each of the smaller gaps need to be expanded to larger ones, resulting in an increase of g_{max}/g_{min} in size or $\log(g_{max}/g_{min})$ extra bits. Similarly, if t_{max} and t_{min} are respectively maximum and minimum gaps for target distribution, we expect at most $\log(t_{max}/t_{min})$ increase in size. Hence the ciphertext will have at most $\log(g_{max}/g_{min}) + \log(t_{max}/t_{min})$ extra bits. Even if we allow the maximum gap to be 2^{16} times more than the minimum gap for each distribution, the resulting increase in ciphertext size will not be more than 4 bytes. Note, that for a TinyOS packet with maximum size 36 bytes (7 bytes signaling data), this is a moderate increase that most probably ensures data to be transmitted in one single packet.

6 Energy Considerations

In this section, we present concrete measurements from our implementation on Crossbow’s MICA2 Mote and show how applying OPES for secure ciphertext comparison helps distributing energy consumption over all nodes in a more balanced way and reduce the energy load in the backbone. Our simulations showed RC5 encryption and decryption each to consume 236 clock cycles, addition (A) operation to consume 4 clock cycles, multiplication (M)⁴ operation to consume 40 clock cycles, division (D) 700 clock cycles, comparison operation 4 clock cycles and finally square-root (S) 1500 clock cycles (all for 4 byte operands). The clock cycle measurements we performed over MICA2 Motes is given in Table 1. For these measurements, the operands were 4 bytes in size.

A plaintext value in OPES is converted to flat value using

$$f = f_{min} + \sum_{j=1}^{i-1} w_j^f + M_i(p - p_{min} - \sum_{j=1}^{i-1} w_j)$$

⁴If a particular computation consumes i additions and j multiplications, we denote its computation cost by $i.A + j.M$

$$M(p) = z(qp^2/2r + p) = zsp^2 + zp$$

By storing precomputed zs and z instead of z and $s = q/2r$, and by computing the mapping function as $M(p) = (zs)p^2 + (z)p$ the computation cost for mapping could be reduced to $3M + A$. Values $(f_{min} + \sum_{j=1}^{i-1} w_j^f)$ and $(-p_{min} - \sum_{j=1}^{i-1} w_j)$ can also be precomputed and hence the computation cost of evaluating f would be $2M + 2A$, given the mapping parameter. A flat value f is then converted to cipher value c using

$$c = c_{min} + \sum_{j=1}^{i-1} w_j^c + (M_i)^{c-1}(f - f_{min} - \sum_{j=1}^{i-1} \bar{w}_j^f)$$

$$M^{c-1}(f) = (-z \pm \sqrt{(z^2 + 4zsf)})/2zs$$

The computation cost for evaluating c , given the mapping parameter, is same as that for f which is $2M + 2A$. We rewrite the mapping function as $M^{c-1}(f) = -1/(2s) \pm \sqrt{1/(4s^2) + (1/(sz))} \cdot f$. By precomputing $-1/(2s)$, $1/(4s^2)$ and $1/(sz)$, the mapping operation would cost $1M + 2A + 1S$. Finally to determine the bucket in which a plaintext p falls and a flat value f falls, by binary search, OPES needs to do at most $\log(m + 1)$ comparison operations ($m + 1$ bucket boundaries can be sorted before storing). Hence the total cost of OPES encryption (adding all previous values) is $8M + 7A + 1S + \log(m + 1)C$ or about 1800 clock cycles. Now assume that an aggregator aggregates n sensed values (that is the aggregator finds the maximum among n values). In the RC5 scheme, all of the n values have to be decrypted before comparison and the result encrypted again. Hence it would consume $236(n + 1)$ clock cycles. Ciphertext comparisons in OPES happens over 8 byte values at the most, as the increase in ciphertext size is at most 4 bytes more than the plaintext size. Hence the total energy consumed by OPES for aggregation is $8n$, as n comparisons are required to find the maximum among n values. For all positive values of n , $8n < 236(n + 1) + 4n$ holds which means energy gains at the aggregator is observed even for very low density networks. If an aggregator aggregates 20 values (the normal density of sensor networks is about 20-40 nodes in the radio range of a given node), the energy savings at the aggregator is close to 5000 clock cycles in the OPES scheme. The fact that any aggregator could be used for aggregation (aggregators need not have the key to aggregate) and that the aggregators consume drastically lesser energy for aggregation when compared to hop-by-hop schemes allows election mechanisms like LEACH [10] to evenly distribute the energy consumption over all the nodes by properly choosing the nodes to either sense, aggregate, forward or sleep at different time epochs. The network backbone life can be extended.

7 Conclusions

End-to-end encryption schemes that support operations over ciphertext are of utmost importance for commercial private party Wireless Sensor Network implementations to become meaningful and profitable. A large class of useful sensor network applications rely exclusively on comparison operations. In this paper we demonstrated that a certain order preserving encryption, OPES [2], if used for secure ciphertext comparison, achieves (1) robustness and increased security due to keyless aggregation and (2) better energy distribution and increased backbone life. The resulting memory, message and computation overhead is shown to be reasonable over MICA2 Motes.

References

- [1] Crossbow technology inc. wireless sensor networks. <http://www.xbow.com/Products/products.htm#Wire>. Accessed in October 2004.
- [2] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. *ACM SIGMOD*, June 2004.
- [3] G. Bebek. Anti-tamper database research: Interference control techniques. *Technical Report EECS 433 Final Report, Case Western Reserve University*, November 2002.
- [4] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. *IEEE Symposium on Research in Security and Privacy*, pages 197–213, 2003.
- [5] J. Domingo-Ferrer. A provably secure additive and multiplicative privacy homomorphism. *Information Security Conference, LNCS 2433*, pages 471–483, January 2002.
- [6] L. Eschenauer and V. D. Gligor. A key management scheme for distributed sensor networks. *Proceedings of 10th ACM Conference on Computer and Communication Security*, pages 42–51.
- [7] J. Girao, D. Westhoff, and M. Schneider. Concealed data aggregation in wireless sensor networks. *ACM WiSe04 - poster, in conjunction with ACM MOBICOM 2004*, October 2004.
- [8] J. Girao, D. Westhoff, and M. Schneider. CDA: Concealed data aggregation for reverse multicast traffic in wireless sensor networks. *40th International Conference on Communications, IEEE ICC 2005*, May 2005.
- [9] H. Haciguemues, B. R. Iyer, C. Li, and S. Mehrotra. Executing sql over encrypted data in database-service-provider model. *Proc. of the ACM SIGMOD Conference on Management of Data*, June 2002.
- [10] W. B. Henzelman, A. P. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, 1(4), October 2002.
- [11] C. Karlof, N. Sastry, and D. Wagner. Tinysec: A link layer security architecture for wireless sensor networks. *Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, November 2004.
- [12] S. C. G. Ozsoyoglu and D. Singer. Anti-tamper databases: Querying encrypted databases. *Proc. of the 17th Annual IFIP WG 11.3 Working Conference on Database and Application Security*, August 2003.
- [13] A. Perrig, R. Szewczyk, V. Wen, D. E. Culler, and J. D. Tygar. SPINS: Security protocols for sensor networks. *Mobile Computing and Networking*, pages 189–199, 2001.
- [14] J. Rissanen. Stochastic complexity in statistical inquiry. *World Scientific Publication*, 1989.
- [15] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.
- [16] A. Weimerskirch and D. Westhoff. Identity certified zero-common knowledge authentication. *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '03)*, October 2003.
- [17] A. Weimerskirch and D. Westhoff. Zero-common knowledge authentication for pervasive networks. *10th Selected Areas in Cryptography (SAC'03)*, pages 73–87, July 2003.