An Energy-Efficient Approach for Provenance Transmission in Wireless Sensor Networks

S. M. Iftekharul Alam
Purdue University
alams@purdue.edu

Sonia Fahmy
Purdue University
fahmy@cs.purdue.edu

***Abstract*—Assessing the trustworthiness of sensor data and transmitters of this data is critical for quality assurance. Trust evaluation frameworks utilize *data provenance* along with the sensed data values to compute the trustworthiness of each data item. However, in a sizeable multi-hop sensor network, provenance information requires a large and variable number of bits in each packet, resulting in high energy dissipation due to the extended period of radio communication, and making trust systems unusable. We propose energy-efficient provenance encoding and construction schemes, which we refer to as Probabilistic Provenance Flow (PPF). To the best of our knowledge, ours is the first work to make the Probabilistic Packet Marking (PPM) approach for IP traceback feasible for sensor networks. We design two bit-efficient provenance encoding schemes along with a complementary vanilla scheme. Depending on the network size and bit budget, we select the best method using mathematical approximations and numerical analysis. Our TOSSIM simulations demonstrate that the encoding schemes of PPF have identical performance with a low bit budget ($\sim$ 32-bit), requiring 33% fewer packets and 30% less energy than PPM variants to construct provenance. With a two-fold increase in bit budget, PPF with the selected encoding scheme reduces the energy consumption by 60%.** [1]

***Index Terms*—provenance; trust framework; probabilistic packet marking; energy-efficiency; sensor networks**

I. Introduction

New micro sensors have enabled wireless sensor networks (WSNs) to gather real-time data from the physical world [1], [2]. Planet-wide sensor networks [3], [4], sensor networks for large-scale urban environments [5], and physical infrastructure systems [6] indicate potential deployments of multi-hop networks consisting of hundreds of sensor nodes. In such networks, data produced by the sensors are collected at the base station and made available to decision makers for further analysis. As the quality of decision making is critically dependent on the quality of transmitted information [5], trustworthiness of information and information-transmitting nodes is important [7]. In a multi-hop network, *provenance* includes knowledge of the originator and processing path of data since its generation. While a few provenance-based trust evaluation frameworks have been proposed [8], [9], they do not investigate *energy dissipation* due to provenance transmission.

Provenance of a data item can be represented as a tree which is embedded as meta-data with the item, and updated along the path used to forward the item to the base station [9]. In this case, every intermediate node carries provenance of length proportional to the hop count between that node and the originator of the data item. In a network with a large diameter (hop count), this increased meta-data length results in an extended period of radio communication and energy dissipation at every intermediate node. We consider a real deployment of a 46-hop network [10] in our simulations, and observe that aggregated energy dissipation of the network increases by 27% when a traditional trust framework is employed. Although large networks can be hierarchically organized [11], they still require a significant number of hops [12], with non-negligible energy usage for provenance transmission.

Provenance encoding and construction is similar in nature to the well-known *IP traceback* problem [13], [14]. IP traceback aims to determine the forwarding paths of spoofed packets in traditional wired networks. Among the many proposed solutions to this problem, Probabilistic Packet Marking (PPM) can be most easily adapted to WSNs [15]. In our previous work [16], we showed that direct application of PPM to WSNs is infeasible since it embeds a single node identifier in each packet, and hence requires a large number of packets to construct the forwarding path. Instead, we proposed a new approach, Probabilistic Provenance Flow (PPF), where a connected subgraph of the forwarding path is probabilistically incorporated into a packet.

In this paper, we extend PPF by designing and analyzing three new bit-efficient provenance encoding schemes that ensure faster convergence of provenance construction in an arbitrarily large multi-hop network. We investigate the selection of encoding scheme and its parameters given a fixed bit budget for provenance encoding. Our simulation results show that PPF with the selected encoding scheme can consume up to 60% less energy than the traditional approach, which significantly increases the network life-time.

The remainder of this paper is organized as follows. We formulate the problem of energy-efficient provenance transmission in Section II. Section III discusses related work. Section IV explains three different encoding schemes under PPF. We discuss the corresponding approaches to decode and construct provenance in Section V. In Section VI, we examine the selection of parameters for one of the encoding methods. We analyze the bit requirements for embedding provenance using all encoding schemes in Section VII. Section VIII presents TOSSIM simulation results. Finally, Section IX concludes the paper.

[1]This work has been sponsored in part by NSF grant CNS-0964294.

II. Problem Formulation

A. Network Model

We consider a multi-hop WSN where changes in topology due to failure or mobility can occur, but are infrequent. We make the following assumptions regarding the network and traffic:

(1) A Base Station (BS) acts as a central command authority and the root of a routing tree. It has no resource constraints and cannot be compromised by an attacker.

(2) Sensor nodes monitor their surroundings and periodically report to the base station or their designated cluster head (if any).

(3) Multiple sensors are used to monitor an event. Within a particular time window, independent observations obtained at cluster heads (if any) or the base station from different sensors are concerned with the same event.

(4) A provenance-based trust management method such as [8], [9] is used in the application layer to evaluate and manage trust in an adaptive manner. More details can be found in [16].

B. Problem Statement

We consider a network of N nodes, where the maximum length (depth) of any forwarding path (tree) is L. Assume that the maximum number of bits that can be used to embed provenance information in a single packet is B. Based on this bit budget, there is an integer $m, 1 < m \leq L$ such that at most m consecutive node identities (that is, $m-1$ consecutive edges) can be embedded into a single packet. We must perform the following three operations:

(1) **Provenance Embedding**: In a forwarding tree $G = (V, E)$ rooted at the base station, each node $n_i \in V$ makes an independent decision whether to embed its identity into the packet, starting a connected sub-graph, with probability p_i. We need to design a provenance embedding method to carry a partial path $P =< n_{i_1}, n_{i_2}, \cdots n_{i_m} >$ into a single packet where $n_{i_j} \in V, 1 \leq j \leq m$ and $(n_{i_k}, n_{i_{k+1}}) \in E, 1 \leq k \leq m-1$. This problem is a simple extension of the edge sampling approach in IP traceback [13].

(2) **Provenance Construction**: At the base station, we must construct the entire provenance tree $G = (V, E)$ by exploiting partial path information collected from a number of received packets, with an upper bound on the number of packets required to construct the provenance.

(3) **Provenance Evolution**: After topological changes, e.g., due to failures or mobility, we must bound the time that it takes to reflect the changes in the constructed provenance.

III. Related Work

A few provenance-based trust frameworks have been proposed to date [8], [9]. These frameworks do not consider energy-efficiency in WSNs. The problem of provenance transmission is related to the IP traceback problem that determines the forwarding path of spoofed packets [17]. IP traceback methods include hop-by-hop tracing [18], [19], out-of-band ICMP traceback [20], and in-band probabilistic packet marking [13], [14]. Hop-by-hop tracing is not well-suited to WSNs due to its large storage requirement. Hot-spot based traceback methods designed for mobile ad-hoc networks [21], [22] store packet information at the nodes, and traceback is performed hop-by-hop to determine the hot-spot where the attacker is located. In our case, provenance information is continuously required at the base station to compute trust scores of descendant nodes. Hot-spot based methods would incur unnecessary delay in trust score calculation. Out-of-band ICMP traceback requires out-of-band communication and increased bandwidth which limit its usability in resource-constrained WSNs.

In this work, we adapt Probabilistic Packet Marking (PPM) since it does not require additional storage or out-of-band communication. PPM assumes static routes which may not hold in our case. Additionally, PPM requires a significant number of packets to construct the forwarding path. Network coding variants of PPM [23], [24] require fewer packets to construct the forwarding path. Network coding approaches, however, have a high computational complexity and increase the length of the packet, as marking coefficients are transmitted with the packet. Cheng et al. [25] determine the optimal marking probability for each node to reduce the number of packets required to construct the forwarding path.

IV. Probabilistic Provenance Embedding

In this section, we present three provenance embedding schemes as part of our Probabilistic Provenance Flow (PPF) approach. All three methods incorporate node identifiers into a packet probabilistically and only differ in how they encode these identifiers.

A. Juxtaposition of Ranks

In the *rank method*, instead of embedding the node ID directly into a packet, $rank(\text{ID})$ (defined in Definition 1 below) of the node is embedded, since every node ID is uniquely identifiable by its rank, and the rank would need fewer bits than the identity.

Definition 1. *Consider $U = \{ID_1, ID_2, \cdots ID_N\}$ as the set of N node IDs. There is a permutation of U, $\sigma(U) = \{ID_{a_1}, ID_{a_2}, \cdots ID_{a_N}\}$, such that, $ID_{a_j} < ID_{a_{j+1}}$, for $1 \leq j \leq N-1$. Rank of any node ID, $ID_i \in U$, denoted as $rank(ID_i)$, is the position of ID_i in $\sigma(U)$.*

Assume that the packet meta-data has space to hold identities of up to m nodes. We use a counter of $\log_2 m$ bits to track the number of already embedded ranks in the packet. Initially, the buffer and counter contain zeroes. Every node n_i decides to start a connected sub-graph with probability p_i. Once it decides to do so, it overwrites the previous information by doing the following: it zeroes out the entire provenance field and then embeds its rank at the beginning of the buffer and sets the counter to one. If a node decides not to overwrite, it checks for empty buffer space using the counter field. If there is space, it adds its rank into the first available slot in the buffer and increments the counter. Fig. 1 shows an example of this method where the buffer space can hold at most three node identities in a single packet.

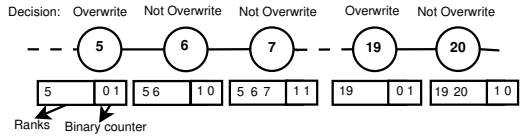


Fig. 1. Provenance encoding using juxtaposition of ranks (numbers inscribed in the circles indicate *rank* of nodes).

B. Prime Multiplication

Our second encoding scheme, the *prime method*, is based on prime multiplication. In a reasonably large network, this method can embed more node IDs within same number of bits (on the average), compared to the rank method. To the best of our knowledge, this method has not been used in any prior work.

TABLE I
BIT REQUIREMENTS FOR MULTIPLICATION OF m NODE IDS, PICKED RANDOMLY FROM THE FIRST f PRIME NUMBERS.

f	$m=3$		$m=4$		$m=6$	
	Avg	Max	Avg	Max	Avg	Max
500	30.80	36	40.76	47	59.82	69
1000	34.25	39	45.35	52	67.02	76
2000	37.82	42	49.84	56	73.64	83
5000	42.33	47	55.83	62	82.45	93

Definition 2. *Let P_n be the largest prime number that is less than or equal to the positive integer n. We define the set of usable IDs, $Q_{P,s}$ where P is a prime number and s is a positive integer:*

$$Q_{P,s} = \{n \in \mathbb{N} \mid 2 \leq n \leq P \text{ and } 0 \leq n - P_n \leq s\}$$

Definition 3. *For any positive integer $n \in Q_{P,s}$, for some P and s, we define two functions:*

- $prime(n) = P_n$ *(the largest prime number that is less than or equal to n).*
- $\text{offset}(n) = n - P_n$.

The prime method is motivated by the idea of using prime numbers as node IDs and encoding a set of IDs through their multiplication which can be uniquely factorized. However, prime multiplication incurs computational and spatial overhead when the participating prime numbers become larger. As shown in Table I, the average number of bits required to multiply m prime numbers increases with the increasing size of the domain of these numbers. This shows the infeasibility of using prime numbers directly as node IDs. Thus, we define $Q_{P,s}$ (Definition 2) to ensure that node IDs will not differ by more than s from their nearest prime numbers where s is referred as the ***spreading factor***. Then, we encode a sequence of node IDs by multiplying their nearest prime numbers and summing up the corresponding offset values (Definition 3). Before describing the details of encoding process, we describe the assignment of node IDs using set $Q_{P,s}$.

1) Node ID Assignment: For a network of N nodes, we pick a set $Q_{P,s} = \{q_1, q_2, \cdots q_z\}$ with the smallest z such that $z \geq N$. An in-place algorithm is used to produce a random permutation of $Q_{P,s}$, $\sigma(Q_{P,s}) = \{q_{a_1}, q_{a_2}, \cdots q_{a_z}\}$ and members of $\sigma(Q_{P,s})$ are assigned to all N nodes sequentially. For example, in an 8-node network, we can pick IDs for the nodes from a random permutation of $Q_{11,7} = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$.

For a given number of nodes (N), the bit requirements for prime multiplication increase the most when $s = 0$, which makes $Q_{P,s}$ nothing but a set of prime numbers that are less than or equal to P ($= N$th prime number). With an increasing value of s, the set $Q_{P,s}$ can contain numbers upper-bound by relatively smaller value of P ($\ll N$th prime number). By tuning s, the largest element (P) of $Q_{P,s}$ can be made close to the total number of nodes (N). This brings about an interesting trade-off: reduction of the number of bits required for prime multiplication versus increase in the number of bits required for summation of offsets. We will investigate this trade-off in Section VI.

2) Encoding Process: To store provenance information, we divide the provenance buffer into two parts: *product* and *offset*. Every node n_i has an ID, say ID_i, that is a member of $Q_{P,s}$ for some P and s. As with the rank method, once a node n_i decides to start a connected sub-graph, it clears the provenance buffer. It then inserts $prime(ID_i)$ into the product part and *offset*(ID_i) into the offset part (Definition 3). If a node n_j decides not to overwrite, it retrieves the values stored in the product and offset parts. It then multiplies the value of the product with $prime(ID_j)$, adds *offset*(ID_j) to the offset, and stores the newly calculated values into the respective parts. Fig. 2 shows an example with $m = 2$.

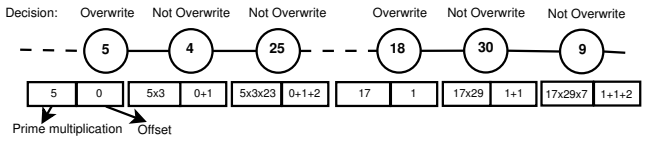


Fig. 2. Provenance encoding using prime multiplication (numbers inscribed in the circles indicate ID of nodes).

We no longer need a counter field to track the number of node identities encoded in the provenance buffer because there is always a unique prime factorization of the *product* part which gives the number of participating nodes.

C. Rabin Fingerprints

The prime method can typically accommodate more node identities (m) than the rank method, but prime multiplication results increase in size as N and m increase. In order to embed more node IDs into a single packet without requiring additional bits and excessive computational complexity, we investigate *Rabin fingerprints* [26].

A *Rabin fingerprint* calculates a near-perfect and space-efficient unique representation of a sequence of bits. For a sequence of bits $n_1, n_2, \cdots n_m$, of length m, the Rabin fingerprint is given by the following expression, where α and M are constant integers:

$$RF(n_1, n_2, \cdots n_m) = (n_1\alpha^{m-1} + n_2\alpha^{m-2} + \cdots + n_m) \mod M$$

The fingerprint of the concatenation of two sequences X and Y can be computed as follows:

$$RF(X||Y) = RF(RF(X)||Y) = RF(RF(X)\times\alpha^{l})+RF(Y)$$

where, $||$ represents concatenation and l is the length of Y.

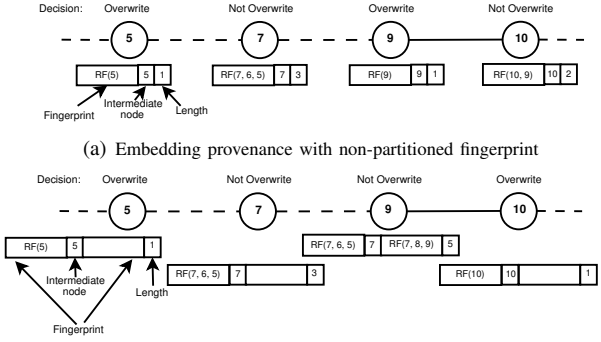
1) Encoding Fingerprints: The partial path traversed by a packet can be considered as a bit sequence of IDs of the nodes on that partial path. We aim at transmitting the fingerprint of that bit sequence sequence instead of transmitting the actual sequence. Every node uses two constant integers α and M to compute its fingerprint. As the packet traverses the path, we could easily compute the contribution of every node to the fingerprint and add it to the contributions of its predecessor nodes on that path. For example, if the packet traverses the partial path $< n_1, n_2, \cdots n_m >$, node $n_i, 1 \leq i \leq m$ has a contribution of $ID_i\alpha^{b(m-i+1)}$ to the fingerprint associated with that path. Here, b is the number of bits required to represent a single ID. However, the incremental sum of these contributions requires a large and variable number of bits in the packet which is undesirable. Hence, we exploit the concatenation property of Rabin fingerprints, which allows any node n_k to compute the fingerprint of node IDs from ID_{n_1} to ID_{n_k} by concatenating its own ID (ID_{n_k}) to the fingerprint value of previous nodes ID_{n_1} to $ID_{n_{k-1}}$. The following equation makes this claim clear:

$$RF(ID_k,ID_{k-1},\cdots ID_1) = RF(ID_k||RF(ID_{k-1},ID_{k-2},\cdots ID_1)) \quad (1)$$

Thus, every node on a path can update the fingerprint without requiring any extra bits as the fingerprint value is always less than the divisor M.

To store provenance information, we divide the provenance buffer into three fields: *fingerprint*, *intermediate node*, and *length*. As in other PPF methods, every node n_i decides to start a connected sub-graph with probability p_i. Once it decides to do so, it clears the buffer and inserts ID_i and 1 into the fingerprint and length fields, respectively. If a node n_j decides not to overwrite, it retrieves the values stored in the fingerprint and length fields. If the length is less than m, it updates the current fingerprint value (say X) by computing $ID_j||X$ and increments the current value of the length field by one. If the newly computed length is less than or equal to m, ID_j is stored in the *intermediate node* field that will aid in decoding the provenance as discussed later.

2) Partitioning Fingerprints: Since we need to exploit previous knowledge about node ordering to retrieve provenance information from the fingerprint (as we will discuss in the next section), transmitting large partial provenance information using fingerprints may not always be advantageous. In WSNs, nodes are vulnerable and error-prone and the routing path may change due to the lossy nature of the wireless medium. This may cause inconsistency between the current and the previously stored ordering among nodes, making the entire fingerprint-based provenance information useless. To mitigate this problem, we divide the fingerprint field into $r(r > 1)$ parts around the $(r-1)$ *intermediate nodes* so that changes in ordering in one part do not affect other parts and changes in ordering can be reflected.



(a) Embedding provenance with non-partitioned fingerprint

(b) Embedding provenance with $m = 5$ and $r = 2$

Fig. 3. Provenance encoding using Rabin fingerprints (numbers inscribed in the circles indicate ID of nodes).

Each part contains fingerprints of at most $\lceil \frac{m+r-1}{r} \rceil$ node IDs where the length field indicates the total number of participating node IDs. For example, assume we wish to embed information about a partial path $< n_1, n_2, \cdots n_m >$ into a single packet with $m = 7$ and $r = 2$. As the packet traverses the network, node n_4 becomes the intermediate node and the first part of the fingerprint contains $RF(ID_4, ID_3, ID_2, ID_1)$ and the second part contains $RF(ID_4, ID_5, ID_6, ID_7)$.

Both parts of the fingerprint are calculated according to equation 1 and the length field indicates the combined length of the both parts. Note that, a non-partitioned fingerprint is a special case where the intermediate node corresponds to the lone fingerprint. Fig. 3(a) depicts an example of the non-partitioned case and Fig. 3(b) depicts the partitioning approach with $m = 5$ and $r = 2$. Note that in adjacent fingerprints, the order in which the fingerprints are calculated is opposite. This ensures checking the integrity of the partitioned fingerprints at no additional cost.

V. Decoding and Constructing Provenance

When a packet is received at the base station, the provenance buffer is examined to retrieve the embedded partial provenance (or path) information. With the rank embedding approach, we can easily extract the embedded identities from the provenance buffer using the length field as the rank of each node ID uses a fixed number of bits. However, with both the prime and fingerprint embedding method, we assume that information about ordering among nodes is known beforehand using a previously constructed provenance graph, $G_{pre} = (V_{pre}, E_{pre})$ (as discussed in Section V-C). Here, V_{pre} is the set of node IDs and E_{pre} is the set of edges among those nodes indicating provenance flow.

A. Decoding Process for Prime Method

We apply a standard prime factorization algorithm over the product part of the provenance buffer to retrieve the set of nearest prime numbers (say, $X = \{X_1, X_2, \cdots X_m\}$). Then, we perform a DFS (Depth-First-Search) over G_{pre} to find all the possible paths consisting of m node IDs whose nearest prime numbers form any permutation of X. We need to modify the DFS to compare the nearest prime number of a particular

node ID with the members of set X while visiting that node and track the offset of the ID when a match is found. With this modification, DFS will take $O(m(|V_{pre}| + |E_{pre}|))$ time and produce all possible sets of node IDs whose nearest prime numbers form a permutation of X. Since every ID does not differ from its nearest prime number by more than s, we can find at most s^m such sets. For each such set, we will sum up the offset values and calculate its difference from the offset value retrieved from the received packet. If the difference is zero, we record the matched set as the retrieved provenance. Otherwise, we conclude that topological changes may have occurred in the network and the previously stored provenance graph may not be up-to-date. Further processing is necessary to determine the provenance, such as checking other combinations of partial paths by considering nodes that are 1 or 2-hop away from the nodes on the recorded path, or triggering the rank ID embedding approach to recover the order. These extensions will be the subject of our future work.

B. Decoding Process for Fingerprint Method

Upon reception of a packet, we retrieve the r fingerprint(s), associated intermediate node ID(s), and the length field indicating the number of participating node IDs. For every intermediate node, $ID_i, 1 \leq i \leq r-1$, we perform a DFS over $G_{pre} = (V_{pre}, E_{pre})$ with following modifications:

- Set ID_i as the root for DFS.
- Search all the nodes within $\lceil \frac{m+r-1}{r} \rceil$-hops away from the node ID_i and compute the Rabin fingerprint using the concatenation property. Computing the fingerprint takes constant number of shift and XOR operations provided that a pre-computed hash table (of size < 4 kB) contains frequently used values.
- After computing every fingerprint of length $\lceil \frac{m+r-1}{r} \rceil$, we compare them to the retrieved fingerprints RF_i and RF_{i+1}. If a match is found, we record the matched path originating from the node ID_i; otherwise, further processing is necessary to determine the exact provenance which we will consider in our future work.

Typically, for every intermediate node, we are searching a smaller portion of the graph G_{pre} using DFS. In the worst case, we may search the entire graph which can be performed in $O(|V_{pre}| + |E_{pre}|)$ time.

False Positive Rate for Fingerprints: Assume that we have at most $x = \lceil \frac{m+r-1}{r} \rceil$ node IDs per partition where m is the total number of node IDs embedded per packet. Since decoding each pair of partitions is independent of others, it suffices to analyze the false positive probability of fingerprinting a path of x node IDs originating from a particular intermediate node ID. Assume that there are n such paths in the provenance graph. Then, the false positive probability $\leq \frac{n^2 x.b}{2^k}$, where k is the number of bits used for fingerprinting and b is the length of the bit representation of one node ID.

If the maximum fan-out of the network is f, then n is upper-bound by f^{x-1} which gives,

$$\textit{False positive probability} \leq \frac{f^{2(x-1)} x.b}{2^k} \quad (2)$$

$$= \frac{b(m+r-1) f^{\frac{2(m-1)}{r}}}{r.2^k}. \quad (3)$$

C. Construction and Evolution of Provenance

With our identity embedding methods, provenance construction is straightforward once we have decoded partial path information from the received packet. After collecting sufficient packets with embedded provenance (i.e., when we have at least one ID from each node), we combine the partial paths to produce the complete provenance graph, $G = (V, E)$. Here, V is the set of nodes and for some $v_i \in V, v_j \in V$, $(v_i, v_j) \in E$ iff (ID_i, ID_j) belongs to some partial provenance encoded in a received packet. However, decoding using the prime and fingerprint approaches needs prior knowledge of the order of nodes. This can be obtained by applying the rank method first. After a configurable period of time (generally greater than provenance convergence time) during which the provenance is constructed using the rank method, the prime or fingerprint embedding method can be employed.

In order to keep node order information up-to-date, nodes utilize the rank approach every $t_{embedding}$ seconds. Thus, any topological changes are reflected in the provenance. Based on the frequency of mobility or failures in the network, $t_{embedding}$ can be adjusted. However, a small value of $t_{embedding}$ will reduce the benefits of applying the bit-efficient prime and fingerprint methods. We are currently considering a reactive approach to trigger the rank approach only when necessary.

D. Complexity Analysis

Decoding using the rank method is straightforward and takes only $O(m)$ time, where m is the maximum number of nodes embedded per packet.

In case of the prime method, we use the General Number Field Sieve (GNFS) algorithm for prime factorization. The asymptotic running time for this algorithm for a b-bit number is $O(\exp((\frac{64b}{9})^{\frac{1}{3}}(\log b)^{\frac{2}{3}}))$.

In an N-node network, node IDs require at most $\lceil \log_2 N \rceil$ bits with the appropriate choice of s and m. Thus, multiplication of m node IDs requires at most $m\lceil \log_2 N \rceil$ bits which makes the complexity of prime factorization $O(\exp((\frac{64m\lceil \log_2 N \rceil}{9})^{\frac{1}{3}}(\log(m\lceil \log_2 N \rceil))^{\frac{2}{3}}))$.

Then, we need to perform DFS which takes $O(m(|V| + |E|))$ and s^m comparisons which take $O(s^m)$ time. We know that s can be approximated as $\frac{N}{\pi(N)} \sim \ln N$ (as discussed in the next section). Thus, the time required to decode provenance from a single packet becomes

$$O(\exp((\frac{64m\lceil \log_2 N \rceil}{9})^{\frac{1}{3}}(\log(m\lceil \log_2 N \rceil))^{\frac{2}{3}}) + mN^2 + (\ln N)^m)$$

which is exponential in terms of $m \ln N$.

In case of the fingerprint method, we need to search the provenance graph and update the fingerprint using concatenation while visiting a node on that graph. This takes $O(|V| + |E|)$ time which is $O(N^2)$ in the worst case.

TABLE II
PRIME GAPS BELOW THE NUMBER n.

n	Prime Gap			
	Observed mean (μ)	Observed stdev	gap_{avg}	$\frac{\mu}{gap_{avg}}$
500	5.29	3.11	5.12	1.03
1000	5.96	3.55	5.82	1.02
2000	6.61	4.50	6.52	1.01
5000	7.48	5.30	7.44	1.01

Regarding the construction of entire provenance in an N-node network, we represent the provenance graph using an adjacency matrix. The total number of edges of the graph can be at most $O(N^2)$. Since edge insertion requires constant time, the worst case complexity for constructing the entire provenance is $O(N^2)$. We need $O(N^2)$ space to hold the provenance graph apart from the space required by the trust framework.

VI. SPREADING FACTOR

The prime method requires two parameters s and P that define the set of node IDs, $Q_{P,s}$.

A. Approximating the Spreading Factor

For a given number of nodes, N, we want to determine the spreading factor, s that minimizes the highest value of $Q_{P,s}$. This value of s depends on the prime gap.

Definition 4. *A prime gap is the difference between two successive prime numbers, p_k and $p_{(k+1)}$, where p_k is the kth prime number. Thus, a prime gap of length n is a run of $n-1$ consecutive composite numbers between two successive primes.*

We use the **Prime Number Theorem** to approximate the average length of prime gaps. The theorem gives an asymptotic form for the prime counting function $\pi(n)$, which counts the number of primes less than some integer n. According to this theorem (proved independently by Hadamard (1896) and de la Valle Poussin (1896)),

$$\begin{aligned}\pi(n) &\sim \sum_{k=0}^{\infty} \frac{k!n}{(\ln n)^{k+1}} \\ &\sim \frac{n}{\ln n} + \frac{n}{(\ln n)^2} + \frac{2n}{(\ln n)^3} + \cdots \end{aligned} \quad (4)$$

It has been shown that summation of the first three terms in equation 4 is a better estimate for $\pi(n)$ (Derbyshire 2004, pp. 116-117). Now, we can approximate the average length of prime gaps below n as

$$gap_{avg}(N) \approx \frac{n}{\pi(n)} \sim \frac{1}{\frac{1}{\ln n} + \frac{1}{(\ln n)^2} + \frac{2}{(\ln n)^3}}.$$

Table II shows the theoretical mean along with empirical mean and standard deviation of prime gaps for different values of n. The last column of this table gives the ratio between empirical and theoretical mean which justifies the approximation above. Assume that P_n denotes the nth prime number. By choosing a spreading factor, s, that approximates to $gap_{avg}(N)$ for some N, we can obtain a set of numbers upper-bound by some prime number $P_{\pi(N)+1} \geq N$. We denote this set as $Q_{P_{\pi(N)+1}, gap_{avg}(N)}$. Due to the high variation in prime gaps with respect to $gap_{avg}(N)$ the cardinality of this set becomes less than N. Assume that using the same spreading factor ($s = gap_{avg}(N)$), we find a set of numbers, $Q_{P_{\pi(N')},s}$ such that $|Q_{P_{\pi(N')},s}|$ is the smallest number greater than or equal to N. Similarly, by choosing some values larger than $gap_{avg}(N)$ for spreading factor s, we can have a set of numbers $Q_{P_{\pi(N'')},s}$, where $|Q_{P_{\pi(N'')},s}|$ is the smallest number greater than or equal to N. Clearly, $P_{\pi(N')} > P_{\pi(N'')}$, which makes the latter set more favorable in terms of bit requirements (as observed in Table I). We use the following optimistic choice:

$$s \geq gap_{avg}(N) \approx \frac{N}{\pi(N)} \approx \frac{1}{\frac{1}{\ln N} + \frac{1}{(\ln N)^2} + \frac{2}{(\ln N)^3}}. \quad (5)$$

TABLE IV
AVERAGE CASE BIT REQUIREMENTS FOR VARYING NUMBER OF NODES AND PER-PACKET NODE IDS WITH s^*.

N	$m=3$		$m=4$		$m=5$	
	s^*	avg	s^*	avg	s^*	avg
500	5	27.82	7	36.16	6	44.35
1000	5	31.25	7	40.53	6	49.9
2000	10	34.5	7	45.14	6	55.81
5000	10	38.99	7	51.11	11	62.4

B. Choice of Spreading Factor

For a network of size N, we first calculate $gap_{avg}(N)$ and by setting $s = \lfloor gap_{avg}(N) \rfloor$, we pick a set $Q_{P_{\pi(N')},s}$ such that $|Q_{P_{\pi(N')},s}|$ is the smallest number greater than or equal to N. Considering the requirements for prime multiplication and summation of offset, we estimate the worst case bit requirements per packet as:

$$B^P_{worst}(s,m) = \log_2(\prod_{i=0}^{m-1} (P_{\pi(N')} - i)) + \log_2(m \times s),$$

where m is the number of node IDs embedded per packet.

Then we increase s by one and determine the corresponding set of node IDs. After calculating the worst case bit requirements for the new set, we compare the newly calculated value with the current one. If the newly calculated set outperforms the current one in terms of worst case bit requirements, we set the new set to be the current one. We continue until the newly calculated set requires more bits than the current one. Table III shows the comparison among the required number of bits for different values of s with varying numbers of nodes and number of per-packet node IDs. For a particular number of nodes and per-packet node IDs, the last column gives the best choice for s (s^*).

VII. BIT BUDGET

A fixed budget of bits (B_{budget}) is available for embedding provenance of at most m nodes within the meta-data of a packet. We give the value of m for our three encoding methods in this section.

TABLE III
WORST CASE BIT REQUIREMENTS FOR VARYING NUMBER OF NODES, PER-PACKET NODE IDS, AND CHOICE OF s.

m	N	gap_{avg}	$s = \lfloor gap_{avg} \rfloor$		$s = \lfloor gap_{avg} \rfloor + 1$		$s = \lfloor gap_{avg} \rfloor + 2$		$s = \lfloor gap_{avg} \rfloor + 3$		$s = \lfloor gap_{avg} \rfloor + 4$		s^*
			P_{max}	Bits	P_{max}	Bits	P_{max}	Bits	P_{max}	Bits	P_{max}	Bits	
3	500	5	613	32	587	33	563	33	547	33	521	33	5
3	1000	5	1307	36	1229	36	1163	36	1109	36	1063	36	5
3	2000	6	2683	40	2477	39	2377	39	2269	39	2213	39	10
3	5000	7	6779	44	6367	43	6079	43	5857	43	5669	44	10
4	500	5	613	43	587	42	563	42	547	43	521	43	7
4	1000	5	1307	47	1229	47	1163	46	1109	47	1063	47	7
4	2000	6	2683	51	2477	51	2377	51	2269	51	2213	51	7
4	5000	7	6779	56	6367	57	6079	57	5857	57	5669	56	7
5	500	5	613	52	587	51	563	52	547	52	521	52	6
5	1000	5	1307	57	1229	57	1163	57	1109	57	1063	57	6
5	2000	6	2683	62	2477	63	2377	63	2269	62	2213	62	6
5	5000	7	6779	70	6367	70	6079	69	5857	69	5669	69	11

TABLE V
CHOOSING (r, m) FOR DIFFERENT BIT BUDGETS IN A 5000-NODE NETWORK WITH $b = 13$ AND $\epsilon = 5$.

r	$B_{budget} = 32$		$B_{budget} = 64$		$B_{budget} = 128$	
	m	x	m	x	m	x
1	2	2	9	9	25	25
2	n.a.	n.a.	7	4	21	11
3	n.a.	n.a.	n.a.	n.a.	16	6
4	n.a.	n.a.	n.a.	n.a.	9	3

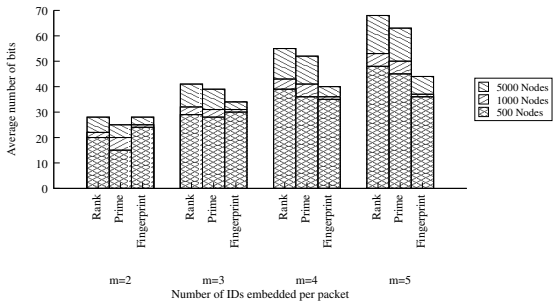


Fig. 4. Average bit requirements for per packet provenance in networks of different sizes. (Stacked data along a single column is given in relative values that need to be added to get the absolute values of each textured bar.)

A. Bit Usage for Rank Method

In an N-node network, bit requirements for the rank method are $B^R(m) = m \times \log_2 N + \log_2 m$, where, the first term on the right hand side indicates the required number of bits to embed m *ranks*, and the second term accounts for the counter that tracks the number of embedded ranks. Thus, we choose the largest m such that $B^R(m) \leq B_{budget}$.

B. Bit Usage for Prime Method

In an N-node network, we can pessimistically pick a value of m such that $B^P_{worst}(m, s^*) \leq B_{budget}$. This does not guarantee the best usage of available bits since prime multiplication of m node IDs does not always need a fixed number of bits (as in the case of rank approach) and $B^P_{worst}(m, s^*)$ can hold more than m node IDs in many cases. Hence, we consider average case bit requirements before choosing an m for a particular bit budget. The average bit requirements per packet are

$$B^P_{avg}(s, m) = \frac{m}{\pi(N')}\Big(\sum_{i=2}^{\log_2 \pi(N')-1} (i * [\pi(2^i) - \pi(2^{i-1})]) + \log_2 \pi(N') * [\pi(N') - \pi(2^{\log_2 \pi(N')-1})]\Big) + \log_2(m \times s).$$

Table IV shows the average number of bits calculated for different numbers of nodes and per-packet node IDs with their corresponding s^*. We choose an m^* such that $B^P_{avg}(s^*, m^* - 1) \leq B_{budget} \leq B^P_{avg}(s^*, m^*)$. For example, in a 1000-node network with 40 bits available for provenance embedding, we choose m^* to be 4 since $B^P_{avg}(5, 3) \leq 40 \leq B^P_{avg}(7, 4)$ (Table IV). This choice provides the opportunity to embed more node IDs per packet on the average.

C. Bit Usage for Fingerprint Method

In an N-node network, the bit requirements for embedding m IDs per packet can be expressed as

$$B^F(r, m) = \begin{cases} b + \log_2 m + k, & r = 1, \\ (r-1)b + \log_2 m + rk, & r > 1, \end{cases} \quad (6)$$

where r is the number of partitions in the provenance buffer, k is the number of bits required for each fingerprint and b is the number of bits required to represent one node ID. We need to choose an m and r such that $B^F(r, m) \leq B_{budget}$. Here, $r = 1$ denotes the non-partitioned case, where the entire provenance buffer can be regarded as a single partition.

Assuming a false positive probability of $2^{-\epsilon}, \epsilon \geq 0$, from equation 2,

$$k = 2(x-1)\log_2 f + \log_2 b + \log_2 x + \epsilon \quad (7)$$

where $m = (rx - r + 1)$.

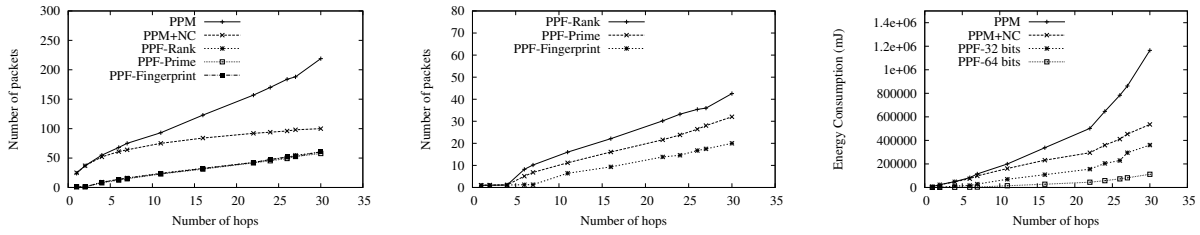
First, we consider the case when $r = 1$, which leads to $x = m$. Then, combining (6) and (7) we have,

$$2(m-1)\log_2 f + 2\log_2 m \leq B_{budget} - b - \log_2 b - \epsilon. \quad (8)$$

Similarly, considering $r > 1$, we have

$$r(2(x-1)\log_2 f + \log_2 x + b + \log_2 b + \epsilon) + \log_2(rx - r + 1) \leq B_{budget} + b. \quad (9)$$

We determine the maximum value of m for different values of $r \geq 1$ using the above two equations. System designers are left to choose the appropriate pair of (r, m) based on the rate of



(a) Provenance construction with 32-bit budget (b) Provenance construction with 64-bit budget (c) Aggregate energy consumption (in mJ)

Fig. 5. Comparison among different encoding schemes of PPF and PPM variants.

link failure or changes, and the average fan-out of the network. In Table V, we consider a 5000-node network with average fan-out $f = 4$, $b = 13$ bits, and $\epsilon = 5$ to show the possible choices for (r, m) where x indicates maximum number of node IDs contained in each partition. Note that fanout should be restricted so that excessive energy consumption at the junction node does not partition the network [27]. We also compare the theoretically calculated average bit requirements of the fingerprint method with the two other encoding schemes in Fig. 4. Clearly, the fingerprint method requires fewer bits than the other methods, as the value of N and m increase.

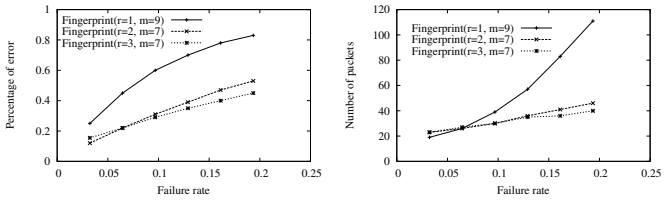
VIII. TOSSIM Results

We conduct simulations using TOSSIM [28] for networks with hop counts ranging from 2 to 30, and number of nodes ranging from 3 to 500. For energy analysis, we use POWER-TOSSIMZ [29] which uses the *micaz* energy model. We do not consider energy consumption related to CPU computations since TOSSIM cannot capture the active CPU time [29]. However, all nodes other than the base station only perform encoding operations which have low computational complexity and are likely to draw insignificant level of CPU power. A base station with no resource constraints can perform the decoding operations, e.g., prime factorization, for a moderate number of nodes in reasonable time. All experiments are performed using a transmission rate of 250 kbps, the default transmission rate of the *micaz* mote, where every data-generating sensor sends data towards the base station every 2 s. The probability for embedding a node ID is $p = \frac{1}{25}$. All results are *averaged over 1000 runs*, and we find the standard deviation to be extremely small.

We consider a 500-node network to compare the performance of the encoding schemes of PPF with two variants of probabilistic packet marking: PPM [13], [14] and PPM with Network Coding [23], [24], as they are the closest to our approach (though they were designed for wired IP networks). We place the same constraint on usable bits (32 bits) for provenance embedding per packet on all schemes.

Fig. 5(a) shows the number of packets required to construct provenance for increasing numbers of hops from a single source to the base station. The results reveal that all three schemes of PPF have identical performance in this case since they can embed only 3 node IDs on average per packet using 32 bits. However, they require at least 33% fewer packets than both PPM variants. The original PPM scheme requires a large number of packets since it embeds a single node ID per packet. Network coding-based PPM (PPM+NC) embeds a linear combination of 3 node IDs in a packet. However, in order to construct a forwarding path of length d hops, PPM+NC converges upon reception of d unique linear combinations of node IDs, whereas PPF only requires $d + 1$ different node IDs.

We perform the same experiment in a 5000-node network with a 64-bit budget in Fig. 5(b). Since TOSSIM does not scale to 5000 nodes, we randomly assign node IDs from a set of 5000 numbers and take the average over experimental results of several TOSSIM runs. We find that the prime method requires fewer packets than the rank method, while the fingerprint method outperforms both in this case. The reason is that with a 64-bit budget, the fingerprint method ($r = 2$) embeds 7 node IDs per packet with a low false positive rate (< 0.001), whereas the prime and rank methods embed 5 and 4 node IDs on average, respectively. Fig. 5(c) compares the aggregate energy consumption for the two PPM variants and the two PPF methods that require the lowest number of packets for the two bit budgets (i.e., prime method with 32-bit budget and fingerprint method with 64-bit budget). PPF with 32-bit budget consumes at least 30% less energy than the PPM variants. PPF with 64-bit budget reduces energy consumption by more than 60% compared to its 32-bit counterpart.



(a) Decoding error of the fingerprint method ($B_{budget} = 64$ bits) (b) Provenance construction using fingerprint method ($B_{budget} = 64$ bits)

Fig. 6. Provenance and associated decoding error with varying rate of link changes.

To observe the effect of partitioning fingerprints, we study the decoding error of the fingerprint method with respect to the rate of link changes. Here, *decoding error* denotes percentage of node IDs that cannot be decoded due to link changes or false positive rates, where *rate of link changes* indicates the average number of link changes per unit time. We artificially introduce link failures and associated path changes which are randomly distributed over a time window of 200 s along a 30-hop path. Fig. 6(a) shows that as link changes increase, the fingerprint method with two partitions ($r = 2$) has a decoding error lower

than the non-partitioned case because of its low sensitivity to topological changes. However, the fingerprint method with $r = 3$ suffers from a false positive rate of about 0.16 with 64-bit budget in this particular experiment, and performs worse than the $r = 2$ case in the presence of low rate of link changes (when decoding error due to link changes is small). With a high rate of link changes, the case of $r = 3$ shows a small improvement over the $r = 2$ case, but a relatively higher false positive rate in a dense network will nullify that improvement (as indicated in Table V). Fig. 6(b) shows the effect of the decoding error in constructing provenance where the fingerprint method with $r = 2$ converges with a fewer number of packets even in the presence of a high rate of link changes and suffers from a negligible false positive rate.

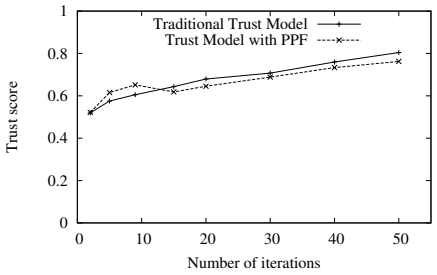


Fig. 7. Trust scores in a 10-hop network.

We also integrate PPF with a provenance-based trust framework to iteratively compute trust scores. Fig. 7 shows that the trust score calculated using PPF evolves correctly as soon as the entire provenance is constructed at the base station. PPF accuracy in trust score calculation is similar to the traditional approach that includes every node ID on the forwarding path in the provenance.

IX. Conclusions

We have presented an energy-efficient provenance transmission and construction approach for large-scale multi-hop wireless sensor networks, based on the idea of probabilistic incorporation of node identities. We adapt the probabilistic packet marking (PPM) approach for IP traceback, and propose three provenance encoding methods with a space constraint on the size of provenance data in each packet. We analyze the suitability of the methods based on the network size and bit budget via mathematical approximations and numerical methods. In contrast to PPM, our proposed approach requires fewer packets to construct network-wide provenance, and significantly reduces the aggregate energy consumption of the network. PPF integration with a provenance-based trust framework on the TinyOS emulator TOSSIM reveals no degradation in accuracy of trust scores. In our future work, we plan to design a reactive approach to accurately reflect topological changes and sensor duty cycling. We will also study how well a complete trust framework can detect and react to different attacks and failure scenarios.

References

[1] D. Butler, "2020 computing: Everything, everywhere," *Nature*, vol. 440, pp. 402–405, 2006.

[2] M. Zuniga and B. Krishnamachari, "Integrating future large-scale wireless sensor networks with the Internet," USC Computer Science, Tech. Rep., 2003, cS 03-792.

[3] K. Fehrenbacher, "A global sensor network launches to fight climate change," January 2011. [Online]. Available: http://www.reuters.com/article/idUS359029730820110112

[4] "Shell to use CeNSE for clearer picture of oil and gas reservoirs," 2009. [Online]. Available: http://www.hpl.hp.com/news/2009/oct-dec/cense.html

[5] A. Doboli and et al., "Cities of the future: Employing wireless sensor networks for efficient decision making in complex environments," SUNYSB, Tech. Rep., April 2008, cEAS Technical Report Nr 831.

[6] "Sensor Andrew at Pennsylvania Smart Infrastructure Incubator." [Online]. Available: http://www.ices.cmu.edu/psii/sensor-andrew.html

[7] J. Ledlie, C. Ng, D. A. Holland, K. kumar Muniswamy-reddy, U. Braun, and M. Seltzer, "Provenance-aware sensor data storage," in *Proc. of Workshop on Networking Meets Databases (NetDB)*, 2005.

[8] X. Wang, K. Govindan, and P. Mohapatra, "Provenance based information trustworthiness evaluation in multi-hop networks," in *Proc. of IEEE GLOBECOM*, 2010.

[9] H.-S. Lim, Y.-S. Moon, and E. Bertino, "Provenance-based trustworthiness assessment in sensor networks," in *Proc. of International Workshop on Data Management for Sensor Networks*, 2010.

[10] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon, "Health monitoring of civil infrastructures using wireless sensor networks," in *Proc. of IPSN*, 2007, pp. 254–263.

[11] A. Arora et al., "ExScal: Elements of an extreme scale wireless sensor network," in *Proc. of IEEE International Conference on Real-Time Computing Systems and Applications*, 2005.

[12] K. Iwanicki and M. van Steen, "On hierarchical routing in wireless sensor networks," in *Proc. of IPSN*, 2009.

[13] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical network support for IP traceback," in *Proc. of ACM SIGCOMM*, 2000.

[14] D. X. Song and A. Perrig, "Advanced and authenticated marking schemes for IP traceback," in *Proc. of IEEE INFOCOM*, 2001.

[15] V. Thing and H. Lee, "IP traceback for wireless ad-hoc networks," in *Proc. of IEEE Vehicular Technology Conference*, 2004.

[16] S. M. I. Alam and S. Fahmy, "Energy-efficient provenance transmission in large-scale wireless sensor networks," in *Proc. of IEEE International Workshop on D-SPAN*, 2011.

[17] A. Belenky and N. Ansari, "On IP traceback," *Communications Magazine, IEEE*, vol. 41, no. 7, pp. 142 – 153, July 2003.

[18] H. Burch, "Tracing anonymous packets to their approximate source," in *Proc. of USENIX conference on system administration*, 2000.

[19] A. C. Snoeren, "Hash-based IP traceback," in *Proc. of ACM SIGCOMM*, 2001.

[20] "ICMP traceback messages." [Online]. Available: http://tools.ietf.org/html/draft-ietf-itrace-04

[21] Y. an Huang and W. Lee, "Hotspot-based traceback for mobile ad hoc networks," in *Proc. of ACM Workshop on Wireless Security*, 2005.

[22] H. Hsu, S. Zhu, and A. Hurson, "A hotspot-based protocol for attack traceback in mobile ad hoc networks," in *Proc. of ACM Symposium on Information, Computer and Communications Security*, 2010.

[23] P. Sattari, M. Gjoka, and A. Markopoulou, "A network coding approach to IP traceback," in *Proc. of IEEE International Symposium on Network Coding*, 2010.

[24] D. Dean, M. Franklin, and A. Stubblefield, "An algebraic approach to IP traceback," *ACM Trans. Inf. Syst. Secur.*, vol. 5, pp. 119–137, 2002.

[25] B.-C. Cheng, H. Chen, Y.-J. Li, and R.-Y. Tseng, "A packet marking with fair probability distribution function for minimizing the convergence time in wireless sensor networks," *Computer Communication*, vol. 31, pp. 4352–4359, 2008.

[26] M. O. Rabin, "Fingerprinting by random polynomials," Center for Research in Computing Technology, Harvard University, Tech. Rep. TR-CSE-03-01, 1981.

[27] H. S. Kim, T. F. Abdelzaher, and W. H. Kwon, "Minimum-energy asynchronous dissemination to mobile sinks in wireless sensor networks," in *Proc. of ACM Sensys*, 2003, pp. 193–204.

[28] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications," in *Proc. of ACM Sensys*, 2003.

[29] E. Perla, A. Catháin, S. Carbajo, M. Huggard, and C. McGoldrick, "PowerTOSSIM z: realistic energy modelling for wireless sensor network environments," in *Proc. of ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, 2008.