Trustworthily Forwarding Sensor Networks Information to the Internet

Olivier Powell¹, Jean-Marc Seigneur² and Luminita Moraru²
1: Patras University, Greece; 2: University of Geneva, Switzerland
Olivier.Powell@gmail.com, Jean-Marc.Seigneur@trustcomp.org, Luminita.Moraru@gmail.com

Abstract
The Internet is soon going to be extended with the information collected from sensor networks deployed in wild remote regions of the world. For example, sensors may be dispersed in the jungle and forward information about the sensed states of the natural ecosystem, such as, humidity, fire detection... However, it is still quite easy for attackers to disconnect the sensors network from the Internet. For example, the sensors usually forward their messages to a base station, the Internet gateway, in a hop-by-hop fashion because they are resource-constrained in terms of energy, the spending of energy dramatically increases with the range of transmission and the attackers may capture intermediate sensors and drop messages rather than forwarding them. In this paper we study how computational trust can be used to mitigate the issue of sinkhole attacks and evaluate our approach on top of the MIX protocol.

1. Introduction
The market of large scale sensor networks is gaining momentum: the deployment of Wireless Sensor Networks (WSN) becomes more and more viable thanks to dropping prices and easier deployment kits. Thus, the Internet may soon be extended with the information collected from sensor networks. A scenario may be that the sensors are dispersed in the jungle and their mission is to forward information about the sensed states of the natural ecosystem, such as, humidity or fire detection, to a far Internet-gateway tower station by forwarding messages between nearby nodes in order to update some Web pages representing the state of the ecosystem to remote users. However, it is still quite easy for attackers to defeat the sensors network purpose. For example, the attackers may capture intermediate sensors and drop messages rather than forwarding them. Another attack may be that the sensors are asked to forward useless information by an attacker and after a while the sensors run out of energy. In this paper we evaluate how computational trust can be used to mitigate the issue of sinkhole attacks when using MIX [10, 9]. In MIX, a sensor can choose to eject a message when all its short-range neighbours have lower energy than itself; to eject means that the sensor increases the power of transmission to be able to reach the base station in one transmission in spite of a far greater loss of energy than when forwarding to a closeby neighbour sensor. This MIX scheme works well when all sensors cooperate. However, in real settings, the cooperation assumption may not be valid. For example, a few sensors may lie about their current energy level to avoid having to forward messages or worse they may not forward messages when asked to do so. In the latter case, these misbehaving sensors carry out an attack commonly called sinkhole attack [8].

The organisation of the paper is as follows. In Section 2, we survey the related work. In Section 3, we present MIX and the model for adjunct trust management. Section 4 describes our evaluation testbed and presents our results. We draw our conclusions and propose future work directions in section 5.

2. Related Work
In this section, we first give an overview of what we mean by computational trust management. Then, we survey how it has been used in the field of wireless networks.

2.1 Computational Trust Management Survey
In the human world, trust exists between two interacting entities and is very useful when there is uncertainty in result of the interaction. The requested entity uses the level of trust in the requesting entity as a mean to cope with uncertainty, to engage in an action in spite of the risk of a harmful outcome. There are many definitions of the human notion trust in a wide range of domains, with different approaches and methodologies, to cite a few, sociology, psychology and economics. Romano’s recent definition tries to encompass the previous work in all these domains: “trust is a subjective assessment of another’s influence in terms of the extent of one’s perceptions about the quality and significance of another’s impact over one’s outcomes in a given situation, such that one’s expectation of, openness to, and inclination toward such influence provide a sense of control over the potential outcomes of the situation” [11].
A computed trust value in an entity may be seen as the digital representation of the trustworthiness or level of trust in the entity under consideration: a non-enforceable estimate of the entity’s future behaviour in a given context based on evidence [13]. A computational model of trust based on social research was first proposed by Marsh [2]. Trust in a given situation is called the trust context. Each trust context is assigned an importance value in the range [0,1] and utility value in the range [-1,1]. Any trust value is in the range [-1,1]. Risk is used in a threshold for trusting decision making. Evidence encompasses outcome observations, recommendations and reputation. The propagation of trust in peer-to-peer network has been studied by Despotovic and Aberer [3] who introduce a more efficient algorithm to propagate trust and recommendations in terms of computational and communication overhead. Such overhead is especially important in sensor networks as any overhead requires more energy spending.

A high level view of a trust engine is depicted in Figure 1. The decision-making component can be called whenever a trusting decision has to be made. Most related work has focused on trust decision-making when a requested entity has to decide what action should be taken due to a request made by another entity, the requesting entity. It is the reason that a specific module called Entity Recognition (ER) [13] is represented to recognize any entities and to deal with the requests from virtual identities.

The decision-making of the trust engine uses two sub-components:

- a trust module that can dynamically assess the trustworthiness of the requesting entity based on the trust evidence of any type stored in the evidence store;
- a risk engine that can dynamically evaluate the risk involved in the interaction, again based on the available evidence in the evidence store.

A common decision-making policy is to choose (or suggest to the user) the action that would maintain the appropriate cost/benefit. In our sensor network application domain, we have to balance ejecting a message or forwarding it based on how much energy has to be spent in each case to successfully reach the base station. In the background, the evidence manager component is in charge of gathering evidence (e.g., recommendations, comparisons between expected outcomes of the chosen actions and real outcomes...) This evidence is used to update risk and trust evidence.

![Figure 1. High-level View of a Trust Engine](image)

Although Jøsang’s “subjective logic” does not use the notion of risk, it can be considered as a trust engine that integrates the element of ignorance and uncertainty, which cannot be reflected by mere probabilities but is part of the human aspect of trust. In order to represent imperfect knowledge, an opinion is considered to be a triplet whose elements are belief (b), disbelief (d) and uncertainty (u), such that:

\[ b + d + u = 1 \]

The relation with trust evidence comes from the fact that an opinion about a binary event can be based on statistical evidence. Information on posterior probabilities of binary events are converted in the b, d and u elements to a value in the range [0,1]. The trust value \((w)\) in the virtual identity \((S)\) of the virtual identity \((T)\) concerning the trust context \(p\) is:

\[ w_p^T(S) = [b, d, u] \]

The subjective logic provides more than ten operators to combine opinions. For example, the recommendation \((\otimes)\) operator corresponds to use the recommending trustworthiness (RT) to adjust a recommended opinion. Josang’s approach can be used in many applications since the trust context is open. In this paper, we base our sensor trust values on this kind of triple and statistical evidence count.

### 2.2 Trust and Security in Sensor Networks

Wireless sensor networks contain hundreds of entities used to collect data from the environment where they are deployed. Usually, each sensor relies on its peers to forward collected data to a central entity, a base station. Sensors are small, low-power devices. Limited in energy, sensors are motivated to have a non-cooperative behaviour when it comes to relaying other sensors packets. They can save power by not forwarding messages received from the neighbours. But selfishness is not the only misbehaviour that a sensor network has to cope with.

An attacker can as well compromise sensors and prevent packets from reaching their destination. Several types of attacks have been identified. A sensor behaving like a sink-hole will drop any packet it receives [8]. In a worm hole attack [8], two colluding sensors create a tunnel between them. The first sensor is situated in the proximity of the base station and...
replays the messages received by the second one. The tunnel is a fast path and will encourage the sensors to use it for routing. In a Sybil attack [4], a sensor claims to have multiple identities. For a routing protocol that uses several paths to the destination, a Sybil attack can advertise one path as several ones. Additionally, it can be correlated with sink hole or worm hole attacks.

A first line of defense is the distribution of private keys to each sensor. But sensors are low cost devices, without tamper proof hardware, thus a captured sensor will permit access to its cryptographic material. Key management schemes [7] try to increase network resilience to sensor capture while maintaining the performance objectives and minimizing the resulting cost of lower network connectivity due to sensors who do not share similar secret keys. There is a trade-off between the energy spent and the cost of used memory for protection and the security level reached [5]. Static keying means that sensors have been allocated keys off-line before deployment, i.e., pre-deployment. The easiest way to secure a network is to give a unique key at pre-deployment time but if only one sensor is compromised the whole network is compromised and it seems viable to extract the key from one sensor as they are cheap and not so well protected (in non-military application scenarios). The second approach is to have pair-wise keys for all sensors on each sensor, which is impractical due to the memory constraints of the sensors. Dynamic keying means that the keys can be (re)generated after-deployment: it creates more communication overhead but stronger resilience to sensor capture. Radio transmission consumes most of the energy spent for security mechanisms, encryption only consumes 3% [5]. Thus, minimizing security transmission is important with regard to energy saving. In [12], they argue that previous approaches relying on keying management and cryptographic means are not suitable for sensors due to their resource constraints or the fact that it is easy to recover their cryptographic material because they are cheap and not fully tamper-proof. Keying material is beyond the scope of this paper that focuses on another way to detect and prevent attacks, that is, to use computational trust management as presented in the previous subsection.

Several mechanisms have already been proposed for mobile ad-hoc networks and wireless sensor networks. They are concerned with making decisions on whether to cooperate or not with their peers based on their previous behaviour. The information used to build the reputation value of neighbours is collected mainly by direct interaction and observation.

Although it is accurate, it requires some time before enough evidence has been collected. In our scenario consisting of static sensors, there is more time to build trust with neighbour sensors since they do not move. It is also the reason that we use a temporary ramp-up counter of 10 messages in the trust metrics of this paper. If recommendations are used, the reputation of the sensors that provide the recommendations has to be taken into account. In this latter case, it may also generate vulnerabilities to false report attacks.

Existing trust models approach differently the previously mentioned reputation building and decision making problems. CORE [13] builds the reputation of a sensor as a value that is increased on positive interactions and decreased otherwise. It takes also into account positive ratings from the neighbours. If the aggregated value of the reputation is positive, the sensor cooperates, otherwise it refuses cooperation. CONFIDANT [1] considers only negative ratings from the neighbours. In order to compute a reputation value, different weights are assigned to personal observation and reported reputation. RFSN [12] uses only positive ratings and models the reputation value as a probabilistic distribution, by the means of a beta distribution model. A sensor will cooperate with the neighbours that have a reputation value higher than a threshold. In [8], they introduce the use of computational trust based on direct observations to mitigate both sinkhole and wormhole attacks. However, their work only covers the mobile ad-hoc network Dynamic Source Routing (DSR) protocol. They cover two trust contexts: TPP, Packet Precision for wormhole; and TPA, Packet Acknowledgment for sinkhole. They combine the two trust contexts. If the sensor is suspected to be a wormhole, the combined trust value \( T \) is \( 0 \). Otherwise \( TPP \) is equal to \( 1 \). \( TPA \) is a counter that is incremented each time a sensor is used to forward a packet and an acknowledgement has been received before a timeout; it is decreased otherwise. The inverse of the combined trust value simply replaces the default cost of \( I \) in the LINK CACHE of the standard DSR protocol. If it is a wormhole the cost is set to infinity.

In this paper, we evaluate the MIX data propagation algorithm for wireless sensor networks. MIX is different than the protocols covered by the previous work on sensor trust management depicted above due to its ejection feature, is impacted by sink hole attacks. In the following section we present which computational trust management model we have chosen.

### 3. Computational Trust Applied to MIX

First, we detail the basic MIX scheme and then how we have added computational trust to MIX.

#### 3.1 The Basic MIX Energy Saving Scheme

The MIX algorithm is a gradient based routing algorithm: when a node needs to send a message it
looks for its lowest neighbour and sends the message to that node. If a node is located at a local minimum, it ejects the message directly to the sink/base station, thus the ejection feature. In MIX, the potential function is such that a node m is considered lower than a node n if:
\[
\text{hop}(m) < \text{hop}(n)
\]
or if
\[
\text{hop}(m) = \text{hop}(n) \text{ and } \text{energy}(m) < \text{energy}(n)
\]
where energy(m) is the energy consumed by node m so far and hop(m) is the hop distance from the mote to the sink. We give a more precise description of the MIX algorithm in the pseudocode of Figure 2.

Algorithm 1 MIX Routing algorithm for sensor n

1. \{When node n gets a message, it will send it to nextMote.\}
2. n = Sensor running the algorithm
3. nextMote = "not defined"
4. for all node m among neighbours of n do
5. if \(\text{hop}(m) \geq \text{hop}(n)\) then
6. nextMote = m
7. else if \(\text{energy}(m) > \text{energy}(n) + \text{hop}(n)\) then
8. nextMote = m
9. else if nextMote is "not defined" then
10. nextMote = m
11. else if \(\text{energy}(m) = \text{energy}(\text{nextMote})\) then
12. nextMote = m
13. else if \(\text{energy}(m) < \text{energy}(\text{nextMote})\) then
14. \{Flip a coin to sort ties.\}
15. nextMote = m with probability 0.5
16. end if
17. end for
18. if nextMote = "not defined" then
19. \{If nextMote is not defined, we need to eject the message to the sink.\}
20. nextMote = sink
21. end if
22. return(nextMote)

Figure 2. MIX routing algorithm for sensor n

It may be noticed that in order to run on a mote, i.e. to make MIX fully distributed, it is required that each node has access to the remaining energy and hop distance of each of their neighbours. Each node is assumed to be aware of its own remaining energy (via its embedded electronics), and each node is assumed to know its own hop distance to the sink. This could typically be computed at the initialisation phase of the network, during which a single flooding occurs from the sink. This implies that every node sends one message (assuming no collisions occur), although optimisation is possible. Knowing the hop distance of neighbours is easy, since it is a constant value in a static network.

However, even in a static network, the energy values change. Therefore, knowing the remaining energy of neighbour nodes, or at least an estimation of this value, requires some extra mechanisms. As explained in [2, 3], this could for example be implemented in a real WSN using standard piggy backing techniques, for example by including in the header of messages the remaining energy and ID of the sender.

3.2 New Computational Trust Scheme for MIX

We make the assumption that at each moment in time, for each node m and for each one of its neighbours n, m knows the number \(s(m, n)\) of messages it has sent to n, as well as \(r(m, n)\) the number of messages m has sent to n which have been received by the sink and \(c(m, n)\) captured by one of the attacker motes respectively. In order to know the value of \(s(m, n)\), node n simply needs to count the messages it sends to m. However, keeping track of \(r(m, n)\) and \(c(m, n)\) may be more difficult in a real WSN (in our simulations, those values are assumed to be available).

One way of implementing this in a real network would be to make the sink send acknowledgements (ACKs) when it receives a message. By broadcasting from time to time a list of (hashkeys) of received messages, for example using a time division multiple access scheme (TDMA), the sink could let all sensors become aware of the safe reception of messages.

This means however that sensors need to store hashables of messages awaiting an ACK and need to listen for the ACKs from the sink, so there is an overhead in energy consumption and memory usage. We would like to stress that the long range broadcasting of ACKs by the sink is not be a problem in scenarios where the sink has plenty of available energy, for example when the sink is plugged on the electrical network.

The TrustMIX algorithm we propose is a generalisation of the MIX algorithm using the following trust function:

\[
\text{trust}(m, n) = \frac{s(m, n) + r(m, n) - c(m, n) + 1}{\text{energy}(m, n)}
\]

The “+1” term in the above definition is just used to avoid confidence dropping to 0 in the case where a single message has been sent and captured (i.e. when \(s = c = 1\) and \(r = 0\)), and it leads to a vanishing term when \(s(m, n)\) tends to infinity. The idea in TrustMIX is that when a node n considers neighbours to which it could forward messages, it will refuse to take into consideration those it distrusts.

More precisely, when a sensor node loops over neighbours to which it would possibly forward a message, the sending node n simply disqualifies any neighbour m with probability \(1 - \text{trust}(n,m)\). TrustMIX is thus a randomized algorithm. Our explanations are detailed in the TrustMIX algorithm below:
Please note that when the network is trustworthy, the algorithm is identical to the original MIX algorithm since the trust between pairs of sensors is 1.

4. Evaluation

We analyse our algorithm through simulations. To conduct simulations, we consider a circle of radius $r$ and randomly and uniformly scatter $n$ sensor nodes in the circle. A sink $S$ is placed at the center of the circle.

Optionally, attacker motes performing a sinkhole attack may be scattered (randomly and uniformly) in the network. We divide time in rounds and each round is divided into two phases. During the event detection phase, we randomly pick a non-attacker mote and make it detect an event by incrementing the size of its message stack by one. During the message propagation phase, each sensor with a non-empty message stack sends exactly one message. The sending mote runs an above TrustMIX algorithm to decide to which neighbour the message is going to be slid, or if it should eject the message directly to the sink. For simulation purposes, we use the following parameters: $r$ is the radius of the circle in which nodes are initially deployed, $d$ is a density factor and $p$ is the fraction of attacker motes added. Given $r$, $d$ and $p$, we set $n$ to $n = (\Pi \cdot r^2 \cdot d)$ and $a = (p \cdot n)$. In order to evaluate the performances of our proposed algorithm, we run in parallel and on the same network topology and same generated events three scenarios. In the first scenario (1), no attackers are placed in the network and the motes run the standard MIX algorithm. In the second scenario (2), the attacker motes are deployed and the MIX algorithm is used again, without adjunct trust management (i.e. the attack will be fully effective). Finally, the third scenario (3) uses our adjunct trust scheme and the motes run the TrustMIX algorithm as a counter-measure to the sinkhole attack.

We observe that the MIX protocol is vulnerable to sinkhole attacks (scenario 2). The three plots of Figure 3 show the result of our simulations when $r$ and $d$ are set to 8 and 4 respectively, and when $p$ takes values in 10, 25 and 50. We observe that even with a relatively small number of attacker motes (left-hand side plot of Figure 3), attacker motes advertising themselves as having plenty of energy and being close to the sink rapidly become attractive to the MIX routing protocol, and soon manage to hijack a large proportion (80 – 90%) of the total traffic.

The trust engine we use builds confidence levels between pairs of nodes. Trustworthy motes are more likely to be selected for message forwarding, whereas attacker motes will be avoided. The aim of the TrustMIX algorithm is to detect and avoid attacker nodes and to run the normal MIX algorithm on the remaining trustworthy motes. Success for TrustMIX would mean that messages get delivered to the sink while preserving the main feature of the MIX algorithm: increased lifetime for the network.

We evaluate below the ability of the TrustMIX algorithm to route messages to the sink in a network under a sinkhole attack (scenario 3) and show that it succeeds in letting messages bypass the sinkholes. The left-hand side of Figure 4 shows the instantaneous percentage of messages reaching the sink for fixed radius $r = 8$ and $d = 4$, and when the percentage of attackers increases ($p = 10$, $p = 25$ and $p = 50$). For $p = 10$, we see that TrustMIX rapidly manages to safely deliver 80% of the traffic to the sink. The proportion of
delivered traffic increases over time, as the trust engine adjusts trust values for pairs of nodes, and soon reaches about 90%.

![Graph showing messages arrival rates and long-term effect of the trust engine.](image)

Figure 4. Impact on sinkhole attacks on MIX

This should be compared to Figure 3, where we see that MIX delivers only about 10% of the traffic safely, even for a small number of attackers ($p = 10$). The comparison of the three curves on the left-hand side plot of Figure 4 shows that the more attacker nodes there are, the more messages get captured and the longer it takes for the trust engine to adjust accurate trust values for pairs of motes. However, even under harsh conditions ($p = 50$, c.f. lower curve of the left-hand side plot of Figure 4), the TrustMIX algorithm is rapidly capable of delivering 50 – 60% of the traffic safely (again, comparing this to Figure 3, where only about 10% of the traffic reaches the sink for the MIX algorithm even for $p = 10$). On the right-hand side of Figure 4, we see that even for $p = 50$ the TrustMIX algorithm manages to route safely about 80% of the traffic, but it needs more time to adjust correct trust values. Summarizing, we see that TrustMIX successfully attenuates the effect of a sinkhole attack on the network by successfully delivering an important fraction of the total traffic.

A preliminary evaluation of the energy consumption overhead of TrustMIX indicates that the average energy consumption of TrustMIX is similar to the one of MIX with only a slightly faster energy depletion of the peripheral nodes, i.e., the farthest nodes from the base station.

5. Conclusion

We have proposed TrustMIX, a fully distributed and simple algorithm for wireless sensor networks which extends the MIX algorithm to be able to mitigate sinkhole attacks. Our two main results are that the TrustMIX algorithm successfully lets most of the data avoid sinkholes and preserves the main features of MIX: increased lifespan of the network.

Future work will analyse parameters which have not been considered in this paper. Such parameters include a finer-grained evaluation of the TrustMIX energy consumption, the deployment of attackers in a non-uniform manner (for example, a region of the network could be attacked or attackers could be deployed in strategic regions, e.g., close to the sink). Also, we have only considered the ratio of delivered to captured messages in the case of uniform events generation, but it would be nice to investigate whether messages away from the sink are more likely to get captured than those close to the sink.

Research supported by the Swiss National Foundation (PBGE2-112864), the Swiss SER (05.0030) and the EU (FP6-IST-004312).

7. References