Secure Web Services for Wireless Sensor Networks

João Guerreiro
Abstract

Secure Web Services for Wireless Sensor Networks

João Guerreiro

Sensor deployments are becoming more and more common nowadays and the ways to access them are becoming more standardized. Indeed, users want to access sensor data via the Internet and without using some complex and unknown protocol; enter Web Services. By observing the typical system architecture for relaying sensor information to the web, we identified out of a large group of security issues a particular one. The issue in question is user privacy.

In this thesis we focus on hiding the activity of a user who queries a sensor deployment, from an attacker that can listen to communications in the neighborhood of the network. Our goal is to generate extra traffic in an intelligent way so that it can effectively mask user activity without draining the energy from the sensors.
Acknowledgments

My first thanks go directly to my parents. They always worried about creating the necessary conditions for me to carry out my studies successfully. I deeply thank them for their love and patience for putting up with me all these years.

Obrigado Mãe. Obrigado Pai.

I would like to thank my supervisor Edith Ngai for her guidance, availability and also for giving me room for my imagination during the thesis work.

Student life is not limited to studying, it is also important to relax and have some fun with friends. I thank my closest friends Guillermo, Mattia, Seif, Thanh and Tian for all the funny adventures, the long conversations and the long walks.
## Contents

1 Introduction ......................................................... 9  
   1.1 Background .................................................. 9  
       1.1.2 Web Services ........................................... 10  
       1.1.3 REST [1, 2, 3] ......................................... 11  
       1.1.4 SOAP [3, 4, 5] ......................................... 13  
       1.1.5 SOAP versus REST ...................................... 16  
   1.2 Thesis Structure ............................................. 17  

2 Related Work ................................................... 18  

3 Problem Description ........................................... 20  
   3.1 System Model .................................................. 20  
   3.2 Terminology ................................................... 21  
   3.3 Periodic Sampling ............................................. 22  
   3.4 Types of Attacks ............................................. 23  
       3.4.1 Active Attacks ......................................... 23  
       3.4.2 Passive Attacks ......................................... 23  
   3.5 Goals ......................................................... 23  

4 Probabilistic Sampling ........................................ 26  
   4.1 Communication Scenarios .................................... 26  
   4.2 Description of the Method ................................... 27  
   4.3 Algorithm ..................................................... 29  

5 Evaluation ....................................................... 33  
   5.1 Setting ......................................................... 33  
   5.2 MAE Comparison ................................................ 34  
   5.3 RMSE Comparison ............................................. 35  
   5.4 Mean Number of Transmissions ............................... 37  
       5.4.1 Periodic Sampling ....................................... 37
5.4.2 Probabilistic Sampling ........................................ 38
5.5 Variance of the Number of Transmissions .................... 42
  5.5.1 Periodic Sampling ........................................... 42
  5.5.2 Probabilistic Sampling ...................................... 43
5.6 Attacker’s Methodology ......................................... 46
  5.6.1 Attacker’s Algorithm ......................................... 46
  5.6.2 Attacker’s Success Rate .................................... 46
  5.6.3 False Positives & False Negatives Rates .................. 48
5.7 Remarks .......................................................... 50

6 Conclusion & Future Work ........................................... 51

Bibliography .......................................................... 53
List of Algorithms

1. Probabilistic Sampling Algorithm .................................................. 30
2. Attacker’s Algorithm ................................................................. 46
List of Figures

1.1 IP Interoperability ............................................. 9
1.2 Web Service invoking other Web Services .................... 11
1.3 SOAP document .................................................. 14

3.1 System model with Web Services ............................ 21
3.2 Intervals and Time Slots ....................................... 21
3.3 Adding the User Query feature to Periodic Sampling ....... 22
3.4 Periodic Sampling with period = 5 .......................... 25

4.1 Two Communication Scenarios in Probabilistic Sampling .... 26
4.2 Two Communication Scenarios in Probabilistic Sampling .... 28
4.3 Increasing Probability Policy ................................. 28
4.4 Only one User Query per Interval allowed .................. 29

5.1 MAE comparison with varying Interval Size .................. 34
5.2 MAE comparison with varying User Queries ................... 35
5.3 RMSE comparison with varying Interval Size ................ 36
5.4 RMSE comparison with varying User Queries ................ 37
5.5 Mean number of transmissions for Periodic Sampling with varying Interval Size .......................... 38
5.6 Mean number of transmissions for Periodic Sampling with varying User Queries .......................... 38
5.7 Gateway transmissions with varying Interval Size using Probabilistic Sampling .......................... 39
5.8 Gateway transmissions with varying User Queries using Probabilistic Sampling .......................... 40
5.9 Sensor transmissions with varying Interval Size, using Probabilistic Sampling .......................... 41
5.10 Sensor transmissions with varying User Queries, using Probabilistic Sampling .......................... 41
5.11 Variance of Transmissions with varying Interval Size, using Periodic Sampling .......................... 42
5.12 Variance of Transmissions comparison with varying User Queries, using Periodic Sampling
5.13 Variance of Gateway Transmissions with varying Interval Size
5.14 Variance of Gateway Transmissions with varying User Queries
5.15 Variance of Sensor Transmissions with varying Interval Size
5.16 Variance of Sensor Transmissions with varying User Queries
5.17 Attacker’s success rate with varying Interval size
5.18 Attacker’s success rate with varying User Queries
5.19 Attacker’s mistakes with varying Interval Size and User Queries
6.1 Better day coverage by combining A and B sensor readings
Chapter 1

Introduction

1.1 Background

1.1.1 IP on Wireless Sensor Networks [10, 11]

Nowadays there is a trend for connecting embedded devices to the Internet using IP. This opens the possibility of intercommunication between IP devices without requiring some sort of translator situated at a network’s gateway. Clearly, following the will of the community is already a strong argument, if we want to be able to easily integrate our network on the Web.

With IP always comes reliable TCP and UDP. So it would be wise to
benefit from using already available transport protocols. Besides already available transport protocols, there are also a set of tools designed for IP; such as firewalls for security.

Other than the interoperability argument, we should additionally consider if it is feasible to have IP on sensors. A. Dunkels demonstrates [8] that it is feasible to fit IP coupled with TCP on resource constrained devices. Since then, further steps have been taken and IPv6 became also supported [9] on sensors; enabling a wider addressing space.

### 1.1.2 Web Services

Web Services are frequently associated with SOAP, but that does not mean it is the only way of building Web Services. After the later appearance of REST, there is now more alternatives. We will now try to define Web Services, while separating that notion from SOAP.

The creation of Web Services can be summarized to an effort of standardization. Companies possessing several different applications, may want them to exchange data easily, whatever the language used by these applications and whatever the platform they are running on. It is desirable then, that the disparate applications use some similar semantics, e.g. XML, to exchange information among themselves. Of course, if companies what to exchange data internally, they might also want to exchange information with external applications that are made available by other companies. Web Services can then, be available in a internal network, but also across the Internet.

It should also be known, that Web Services are primarily made for interactions between machines and not with users. Web Services are not equivalent to web pages; these pages can be created to give a more pleasant GUI \(^1\) for the client to visualize information provided by Web Services. By what has been said until now, we can conclude that Web Services can invoke other Web Services. We can give an example of the usefulness of this property.

Imagine you have a system that helps you plan your vacation. Consider that this system provides you a budget for the total cost of your vacation. That system will invoke three services; one to obtain the price of flight, another for the price of the hotel and yet another for the price of a rented car. By combining the results of these three services the system is providing a service by its own.

---

\(^1\)Graphical User Interface
1.1.3 REST [1, 2, 3]

In this section we will present the most important notions that characterize REST. There is a lot of literature and blogs with different opinions about what REST really is. In this vast ocean of sources, it is easy to get lost; we will try to make things clear to the reader while keeping it short.

1.1.3.1 REST in theory

REST stands for Representational State Transfer and is simply a software architectural style. This architecture was presented for the first time on Roy T. Fielding’s PhD thesis (fifth chapter). The motivation for developing REST was to enable architectures to evolve without any adversity. To enable this property, REST prescribes the use of uniform interfaces between components. By adopting this measure, components can change independently over time, without observing any prejudice in the future. Another good side effect is that we can have a better overview of the whole architecture.

But how can we define “uniform interface”? REST lays down four principles that need to be respected in order for a interface to be considered “uniform”. These are:

- **Identification of resources**: A resource is just a concept. In the web, resources will be identified via URI’s\(^2\). Though we send messages

\(^2\)Uniform Resource Identifier
to resources we will not modify them but instead we will work with their representations. Resources and their representations are two distinct concepts. As an example, when a client requests a resource, its representation will be sent back to the client.

- **Manipulation of resources through representations**: Representations are the current state of a resource. In practice it's a sequence of bytes plus metadata to describe those bytes. It can also include metadata that describes metadata (used for message integrity). If we want to modify a resource we have to do it through the representations instead of modifying the resource itself. By doing this, we are applying the concept of “Information Hiding” from software engineering.

- **Self-descriptive messages**: This is one of REST’s key features; it implies that communications between a client and a server must be stateless. Stateless in this case means that a request sent by a client to a server should be understandable by the latter, without the server having the need of consulting some extra information stored on its side. Basically, messages sent between components should be easily understandable by the receivers.

- **Hypermedia as the engine of application state**: If we analyze this constraint by parts we will be able to understand more easily what it actually means. First, hypermedia is a generalization of the word “hypertext”, but while hypertext stands for displayed text that can refer to other text, hypermedia extends this notion for every kind of media. Secondly “application state” is synonym of “session state”, meaning the state where a user is at, when he is trying to accomplish a task. So this constraint means that the user only needs to navigate from a resource to another in order to reach his desired state (reach his goal). This navigation is done thanks to URI’s.

Now that we know the preconditions for having an uniform interface between components, we can define further constraints imposed by this architectural style.

Since components are separated by an uniform interface we can consider that clients and servers are separated according to this principle. This means that a server won’t have to worry about retaining the user’s state and a client does not have to worry about what the server stores. To summarize; a client does not have to adapt its design according to the server and vice-versa.
An additional constraint is that a REST architecture is designed as a layered system. A component cannot see beyond the layer he is communicating directly with; it may be an intermediary or the end server for example. This constraint enables components to keep their interfaces without changing them; even if new intermediaries are added between a component and the end server.

In REST, the use of caching is prescribed, so that we can avoid some useless network interactions. A requirement is that every response to a request should contain information that says if that response is cacheable or not.

Any service that conforms to the set of hard constraints described above is classified as RESTful.

1.1.3.2 REST in practice

History says that HTTP\(^3\) was developed alongside REST. In fact by using the REST abstraction, some suggestions were made for how HTTP should be designed. So it is no surprise that by using HTTP, a Web Service is implicitly following REST’s constraints. So nowadays developing a RESTful Web Service, boils down to using HTTP as its transport protocol. REST is not strict about the format of the data that needs to be transferred, but some widely popular used formats are XML\(^4\) and JSON\(^5\).

The majority of RESTful Web Services makes heavy use of the GET method (to retrieve information), but it does not mean that the other HTTP methods (PUT, POST, DELETE) should not be used.

A practical example of this trend is Google’s search engine. Imagine you are searching for information regarding the REST topic, then you can just use the URI:

\[
\text{http://www.google.com/search?q=REST}
\]

By doing this, you are using the GET method to retrieve the results of a search that has as “REST” as parameter in the query string. All the necessary information to interpret the request is on the URI.

1.1.4 SOAP [3, 4, 5]

This section will describe the SOAP protocol both in theory and how it is commonly employed in reality. Though the theory of SOAP gives some free-
dom in the usage of this protocol, we will see that in practice it is employed in a recurrent manner.

1.1.4.1 SOAP in theory

SOAP stands for Simple Object Access Protocol. It is used to define the format of the messages that are sent between components. We will say that components exchange between themselves SOAP documents. These documents are basically XML files that follow a certain layout. Every SOAP document should contain the following elements: envelope, header, body and sometimes fault.

![SOAP Document Diagram]

- **Envelope**: This element is the root element of the XML file, it actually defines the file as being a SOAP message. All the other elements are child elements.

- **Header**: This is the first sub-element of Envelope. It contains information about the message itself. The kind of information that you should find in a header is related to routing and security.

- **Body**: This is the part of the envelope that contains the real message that needs to be transferred to the final receiver. It is the second sub-element of Envelope.
• **Fault**: This element is a sub-element of the body. It only appears if there is an error during the transfer of the message. The fault element contains other sub-elements that provide more information about the fault itself; an explanation of the fault, which kind of fault, who caused the fault.

As we said this is the layout of a SOAP document. But this does not mean that the document should have a structure for whatever that is inside the main elements described above. So inside the XML file representing the SOAP message, there should be no DTD\(^6\) for example.

Concerning the mechanism for transporting SOAP messages; there is none defined. The developer can choose to send a SOAP document via HTTP, SMTP\(^7\), FTP\(^8\), ... The reason for this freedom, is that all the required information about the message is in the Header element that is already situated inside the XML file.

SOAP is a protocol for exchanging messages between a requester and a provider of a Web Service, but how can the requester discover Web Services? To advertise Web Services, there is a UDDI\(^9\), which is a directory containing all the services that can be provided. Inside the UDDI, we will find descriptions and locations for each service; each of the entries in the UDDI is written in a XML file that follows a format called WSDL\(^10\).

### 1.1.4.2 SOAP in practice

We have said that SOAP messages can be transported using a variety of protocols. In fact, due to its high acceptance, the major protocol used for transporting SOAP messages is HTTP.

Concerning the HTTP methods that are used; SOAP advises the use of GET and POST for two cases. The HTTP GET should be used when a party wants to retrieve information, without modification of its content. If the information is modified then the use of HTTP POST is advised.

Unfortunately the advice is not followed and almost all the time you will observe the systematic use of HTTP POST, even for simple retrieval of information. The real meaning of the desired operation (retrieval or modification) can be found inside the XML document.

\(^6\)Document Type Definition  
\(^7\)Simple Mail Transfer Protocol  
\(^8\)File Transfer Protocol  
\(^9\)Universal Description Discovery and Integration  
\(^10\)Web Services Description Language
1.1.5 SOAP versus REST

It is difficult to compare REST and SOAP for a few reasons; it is a very live debate about which of these methods is the best, both have strong proponents. Finally, myself I don’t have extensive real life experience using SOAP, so it is difficult for me to judge wetter or not it is a good protocol. Reading through some opinions [6, 7] and looking back at the definitions, I came with some conclusions.

Firstly I would like to make a remark. Although the definitions of REST and SOAP differ, they have something in common; in practice they adopt the HTTP protocol for transferring data. SOAP does not force the use of HTTP, so we could argue if wetter or not HTTP is the best protocol for transporting data. I consider that we should stick to reality; the fact is that REST and SOAP are used in conjunction with HTTP.

One can say that the theory behind REST is quite abstract and it is meant to be so, since it is an architectural style, opposed to a protocol like SOAP. However, in the other hand, the way that RESTful services are implemented seems quite easier than SOAP. Some might not agree with this statement arguing that there are good tools for supporting development in SOAP. But one must admit that if you develop in SOAP, you will have to juggle with several libraries; we can say that REST has a smaller learning curve than SOAP, when it is applied in practice.

While SOAP forces the use of XML, REST gives freedom of choice. This aspect is critical in Wireless Sensor Networks due to their limited resources. By selecting REST, we can choose to adopt a lighter format for exchanging data. Indeed, SOAP is too much verbose.

An advantage of REST over SOAP, is the readability. One can understand more easily a HTTP GET request than a request in SOAP. The over usage of HTTP POST in SOAP makes it extremely difficult for someone to understand the effects of such request; does it delete a representation, update it ? To fully know the real effects, one has to dig in the XML file, while in REST the HTTP GET is quite explicit.

Regarding the description of web services, WSDL 2.0 can be used to describe both SOAP and RESTful Web Services; this was not the case with WSDL 1.1. So it removes the problem of how to describe the parameters that can be passed to a RESTful service.
Finally it seems that SOAP is aimed at creating complex services unlike REST. Since it might be difficult to model every possible service with just the HTTP methods. I would argue that REST is quite suitable in our case. The services that sensors can provide are quite basic; they include reporting values for current temperatures, humidity, detection of movement and others. We can expect that the URI’s for accessing resources will not be too long. So REST is certainly more than enough for the task.

1.2 Thesis Structure

This thesis is organized in the following way. Chapter 2 presents my search for a specific topic of interest and how I got there. Chapter 3 explains what are our objectives for this thesis. Chapter 4 presents a method that I named Probabilistic Sampling, in order to reach the set objectives. In chapter 5 we evaluate our method by analyzing several plots. Finally chapter 6 is dedicated to a conclusion of our work and also some extensions or modifications that can be done.
Chapter 2

Related Work

The starting point for this thesis was another Master thesis written by Dogan Yazar [14], which later on originated an article [15]. The work by Dogan Yazar focused on integrating sensors in the internet by the use of a simple interface; the solution proposed was a REST [1, 2, 3] framework. This work demonstrated the feasibility of using web services for sensors. As the author pointed out at the end of his thesis, one aspect was left out; security. Indeed once the sensor is exposed to the internet [16] it will be under the threat of various possible attacks.

After identifying security [17] as an interesting topic and reading about Web Services, I started wondering how we could provide secure Web Services on sensors [18]. The first question I asked myself was if sensors should really be directly connected to the internet or have some kind of intermediary to regulate who has access [19] to the sensors. Some reflection led me to the conclusion that a gateway is necessary between the internet and a sensor network, for reasons of caching.

Another of my first interests was to know if security mechanism already available on the internet could also be applied in conjunction with sensors. Two popular mechanism are IPSec [20] and HTTPS. Due to its more popular usage, especially in e-commerce, I got interested in HTTPS. Searching through papers, I learned about a project called Sizzle [21], which implements a HTTPS stack in TinyOS\(^1\). The cryptography used in Sizzle is Elliptic Curve Cryptography, which has recently gained the attention of the research community, because of it advantages [23, 22] over RSA in key size for obtaining the same level of security. My interest then shifted, I wanted to find out if there was a some security layer that had been proposed for ContikiOS, which is a operating system for sensors that gained a lot of popularity in Europe.

\(^1\)It is a popular operating system for embedded devices
The reason I concentrated on ContikiOS also came from the fact that there had already been a security layer proposed for TinyOS; TinySec [30].

I found about ContikiSec [24], which does propose a security layer for Contiki and using AES as the main encryption scheme. Later on I found out that work is underway, to integrate Elliptic Curve Cryptography in Contiki.

Finally I saw that there was nothing to my interest to do within cryptography, so I researched some privacy problems [25, 26] and ended up towards the privacy problem of the user, specifically the anonymity of its traffic [27, 28, 29].
Chapter 3

Problem Description

In this chapter we will identify what problem we will be tackling and why it is interesting to do so.

3.1 System Model

Nowadays, Web Services principles are used more and more to request data from embedded devices; one such example is the project ”Energie Visible”\(^1\). In ”Energie Visible”, embedded devices called ”ploggs” communicate electricity consumption readings through JSON files. When requesting data from a sensor network it is a good measure to have a gateway in between to cache some results, otherwise one sensor will not be able to handle the numerous requests. Therefore, taking these two observations in account, we can reach a system model for using Web Services in sensor deployments. This system model is portrayed in figure 3.1.

As we can see in the figure 3.1, the user is provided with a RESTful interface for requesting data. In the first step by employing its web browser, the user requests the temperature of sensor A. This request is then passed on by the gateway to the sensor A. On step 3, the sensor replies with a XML or JSON file that contains the answer. A typical JSON document containing the answer would look like:

```json
[{
  "sensorName": "A",
  "currentTemperature": 17.5,
  "currentTime": "13:53:04"
}]
```

1[^1]: http://www.webofthings.com/2008/12/23/are-you-energy-efficient/
3.2 Terminology

During this thesis you will often hear about *intervals* and *time slots*. Let's define their meaning.

If we consider that sensor readings can be taken during a whole day, we divide the day in intervals of time of equal size. In figure 3.2 we divided the day in intervals of size five. The interval size is actually a parameter that can be chosen by a user; the higher the interval size, the less data reports will be collected.

In this case, since the interval size is five, then that means that each interval contains five time slots. A time slot is a period of time, where only one data report can be made to the gateway and no more than that; if a user queries a sensor but data had already been collected in that time slot, then the user will retrieve a cached value. A time slot lasts between two and three minutes, the reason for this time is due to the data that we used in this
thesis; it corresponds to the time separating each sample collection. More
details will be provided in section 5.1.

3.3 Periodic Sampling

A typical way of collecting data from a sensor deployment is using what I call Periodic Sampling. In this method, data is reported to the gateway periodically. The data can either be requested by the gateway to the sensor(s) or reported autonomously by the sensor(s). Afterwards, the data is saved so that when the user requests information, the data will be retrieved from a cache.

This method is adequate for data collection, but it does not fulfill the desires of a user that has real time requirements. If instead of waiting lets say every ten minutes to have fresh information, the user could wish to be able to query the sensor(s) at a given time instead of waiting another ten minutes. One possibility then would be to enable this feature, but by doing so, we are introducing also a security flaw; indeed the privacy of the user can now be violated.

Consider figure 3.3. In the first line we consider the normal scenario for Periodic Sampling with an interval of size five, where after every four time slots, a new sensor data report is made. In the second line we decided to add the possibility for the user to query the sensors; what happens is that an attacker in the neighborhood of the sensor network that is listening to communications can easily tell that the extra data report in the second interval is not a periodic one. Thus it is easy to identify user activity.

![Figure 3.3: Adding the User Query feature to Periodic Sampling](image)

Figure 3.3: Adding the User Query feature to Periodic Sampling
3.4 Types of Attacks

Attacks to a system can be classified in two categories; active and passive attacks.

3.4.1 Active Attacks

An active attack consists of "messing" directly with the system itself. It can be the interception of messages, their decryption and then modification, it can also be jamming communications, or overloading the system with requests. There are in fact, numerous kinds of attacks [32] that fall into this category.

3.4.2 Passive Attacks

In contrast a passive attack is not focused on attacking directly the system itself, but instead it focuses on doing traffic analysis [13, 33]. In our case, by listening to the sensor communications the attacker could infer at what time of the day a user is absent from his home, or determine points of interest due to a higher number of queries directed to a specific sensor.

A countermeasure to passive attacks is generating dummy traffic. The question is how should that traffic be generated. In this thesis I present an approach for defending against passive attacks, thus preserving the privacy of the user. I call this approach Probabilistic Sampling.

3.5 Goals

This thesis has the goal of proposing a system model that contains these main features:

- Provide the user with the ability to collect sensor data on demand. This way the user has access to fresh information instead of waiting for the start of next period’s data collection.

- Provide security against passive attacks. We do not want an attacker listening to communications near the sensor network, to easily identify user activity; which data collection requests were originated by the user.

- We also want to mask unusual events. An unusual event is defined by the system as something that needs to be reported urgently. As
example, consider there is a fire inside a house, this kind of event needs to be reported immediately; the user cannot wait until the next period arrives, to be notified about it. As the name suggests, the probability of this kind event occurring, is extremely low.

- We want that by adding security, we do not skyrocket the number of data transmissions. An easy way to mask user activity is by collecting data at all times, which is not wise in terms of sensor energy savings, we want to find a smarter way of doing it.

- At the end of the day the gateway interpolates the data and we want the error committed to not be much higher than the error that we would obtain if we used Periodic Sampling.

This is a non-trivial problem since the user’s behavior is unpredictable, so we have to adapt the number of data reports according to the user activity, so that the attacker has a harder time. An important fact is that if we try to minimize the number of data collection times, we also need the samples to be well distributed across time; if they are for example all collected at the beginning of the day, when we interpolate the data for the whole day, the interpolation error will be huge.

In figure 3.4, we compare an interpolation with the real data collected during one day. The x-axis represents the number of the sample. Since we used Periodic Sampling, the samples collected are well distributed across the day, thus the interpolation error is very low; we can see by how similar the two curves are.

One can study user patterns and reproduce it on different days to fake user activity, but still the attacker gains knowledge of how the user’s activity is scheduled for a typical day.
Figure 3.4: Periodic Sampling with period = 5
Chapter 4

Probabilistic Sampling

4.1 Communication Scenarios

First and foremost, it should be mentioned that in Probabilistic Sampling there are two communication scenarios, as it is portrayed in figure 4.1.

![Figure 4.1: Two Communication Scenarios in Probabilistic Sampling](image)

In the first scenario, a query is initiated by the gateway and the sensor replies with a JSON document. This scenario happens when a user decides to query the sensor or when the gateway generates a query on its own; in order to mask user activity.

The second scenario consist of a simple report of data from the sensor, without any request from the gateway. This scenario occurs when an unusual
event happens or when a sensor generates a data report on its own, by the use of our method.

These two communication scenarios exist in order to mask user activity and unusual events.

One can then guess that for our system to work, we need to have the Probabilistic Sampling method running on the gateway to generate queries, but also in every sensor in order to generate sensor reports.

4.2 Description of the Method

The inspiration that I got for devising this method, came from the CSMA\(^1\) protocol. Specifically, CSMA is a MAC\(^2\) protocol that exists in different flavors. The type of CSMA that caught my attention was the p-persistent one, where the p stands for probability.

In Probabilistic Sampling the main idea is to confuse the attacker by reporting data in a non-deterministic way, by using probabilities for transmitting data.

Our method possesses three main policies:

- The use of probabilities for reporting data.
- If too many time slots passed without any data report, then take appropriate action.
- Allow only one user query per interval.

What we mean by using probabilities for reporting data is that at every time slot there will be a certain probability of reporting data, both from a gateway query or by an autonomous sensor report. Both gateway and sensors will have the same probability of initiating a data report. This probability is \(\frac{1}{\text{Interval size} \times 2}\), so that the aggregated probability that one of them initiates the data report is equal to \(\frac{1}{\text{Interval size}}\).

The Interval Size is a parameter chosen by the user; the higher the value of this parameter, the less data reports there will be.

The idea behind this whole policy is that we want to have at least one data report per interval, so that the final interpolation can keep a level of accuracy similar to the Period Sampling approach.

---

\(^1\)Carrier Sense Multiple Access

\(^2\)Media Access Control
Figure 4.2 shows an example where the user chose the interval size to be five, thus there will be five time slots per interval. Consequently, the aggregated reporting probability at each time slot will be \( \frac{1}{5} \).

Regarding the second policy, consider the situation on the first line of figure 4.3. Though we manage to report data at least once in each of the two intervals, there are too many time slots in between without any report. Thus the quality of the interpolation will decrease. So what we do in the second line, is that we adopt a policy of counting the number of time slots without reports. Once we reach the fifth time slot, we highly increase the probability of reporting and we keep that probability until the report actually occurs. When this happens, we can reset the probability to its original value.

Note that we increase the probability, but do not force a report with a 100% chance, otherwise, after some traffic observations, the attacker will be able to completely identify some of the generated reports.

The third policy advocates that a user should only be able to query once,
every time slot. In the first line of figure 4.4, since the user queried three times in a row in the first interval, the attacker can identify a higher activity in that interval, when comparing to the others. It is reasonable to adopt this kind of policy, since the user does not gain much insight by collecting three successive temperature readings.

When a user query occurs, then we have at least one data report in the current interval, thus we can set the probability of generating reports to zero. We adopt this extra policy in order to avoid the situation in the first interval of the second line; after the user query, if the reporting probability is not set to zero, other reports can be made, thus raising the suspicion of an attacker.

Unfortunately there is one situation that we cannot avoid; the one in the second interval of the second line. If auto generated reports occurred before the user query, then we cannot mask the higher activity of that interval. Because of this fact, we can say that our method can never be completely secure against traffic analysis.

Figure 4.4: Only one User Query per Interval allowed

4.3 Algorithm

I have written a Matlab script that simulates the interactions between a gateway and one sensor. Since the algorithm takes too much space, I try to summarize it below, by explaining the main ideas behind it, in a detailed fashion.
Algorithm 1 Probabilistic Sampling Algorithm

Require: Interval Size, #(User Queries), #(Unusual Events)

1: probabilityDummy = \frac{1}{Interval Size \times 2}
2: probabilityUnusualEvents = \frac{#(Unusual Events)}{#(Time Slots)}
3: probabilityUserQuery = \frac{#(User Queries)}{#(Time Slots)}

4: for i = 1 to #(TimeSlots) do
5:   if Time Slot is beginning of new Interval then
6:     Reset value of probabilityUserQuery
7:     Reset value of probabilityDummy
8:   end if
9:   Generate Sensor Report with Probability = probabilityDummy
10:  Generate Gateway Query with Probability = probabilityDummy
11:  Generate Unusual Event with Probability = probabilityUnusualEvent
12:  Generate User Query with Probability = probabilityUserQuery
13:  if No data is collected in current time slot then
14:     zerosCounter = zerosCounter + 1
15:     if zerosCounter \geq Interval Size then
16:        probabilityDummy = \frac{Interval Size}{Interval Size \times 2}
17:     end if
18:   end if
19: else
20:   if More than one event scheduled for this time slot then
21:      Select one or two events
22:   end if
23:   Generate the event
24:   Save the sensed data
25:   zerosCounter = 0;
26:   Reset value of probabilityDummy
27: end if
28: if A User Query was generated then
29:   probabilityUserQuery = 0
30:   probabilityDummy = 0
31: end if
32: end for
This algorithm requires three parameters; the interval size, a number of User Queries and a number of Unusual Events. We explained previously the meaning of the interval size parameter.

The second parameter is the number of User Queries, this parameter is divided by the total number of time slots in a whole day. By doing this we obtain the probability of generating a user query at every time slot; if the value of this parameter is 45 and the total number of time slots is 675, then at every time slot we will have a $\frac{45}{675}$ chance of generating a user query.

For the third parameter, we are provided with a number of unusual events. As for the User Queries parameter, we divide this number by the total number of slots in a day, to calculate the probability of generating an unusual event in a time slot.

The probabilityDummy stands for the probability of the gateway requesting data from a sensor and also for the probability of the sensor reporting data on its own to the gateway. Indeed, both gateway and the sensor have the same chance of generating dummy reports. Dummy reports are not actually that dummy, since they possess valuable data for a later interpolation. I wanted that a dummy would be generated in an interval with a chance of $\frac{1}{\text{Interval Size}}$, so if both gateway and sensor have a probability $\frac{1}{\text{Interval Size} \times 2}$, then the chance that one OR the other reports data is $\frac{1}{\text{Interval Size} \times 2} + \frac{1}{\text{Interval Size} \times 2} = \frac{2}{\text{Interval Size} \times 2} = \frac{1}{\text{Interval Size}}$.

The for loop is for going through all the time slots of the day. At the beginning of a new interval, we reset the probabilities to their initial values.

As you can remember, a strategy adopted by this algorithm is to check if there have not been data reports for some time, if it is the case, then we drastically increase probabilityDummy, so that the interpolation error does not grow too much. This can be seen on lines 14 to 18 of the algorithm.

The meaning of lines 20 to 21, lies on the fact that we consider timing problems. If both a gateway and a sensor want to generate dummies at the same time slot, then either one of them is faster than the other and so the slower one cancels its dummy generation, or none is faster than the other and so both happen.

Lines 28 to 30 come from the fact that if a user query happens during an interval, then no more reports should be made during that interval. We adopted this strategy for security reasons. We also mentioned beforehand that it would not be very interesting for a user to request data, time slot after time slot, since he would not gain much quality of information [12]. We found a compromise; we don’t want the user to wait too long to get fresh
data but we disable him from overloading an interval with queries, which would make the attacker’s job too easy.

Again, I would like to remind the reader, that the above algorithm simulates the interactions between a gateway and a sensor. Our Probabilistic Sampling method needs to be deployed on both parties; gateway and sensor.
Chapter 5

Evaluation

5.1 Setting

To conduct an evaluation of our algorithm, we stress the importance of using realistic data. So, we used data collected by one of the EPFL’s\(^1\) sensors (station ID 1) deployed in their campus, this data is from the day 17/06/2008 and it covers the whole day. There are in total 675 collected samples, which are separated in time by between 2 and 3 minutes. The times that separate each sample are probably due to the latency of communications. So each collected sample is valid for a time between 2 and 3 minutes, this length of time is what we called previously as time slots, only one data report can be collected during a time slot and no more than that.

To do our evaluations we run a Periodic Sampling algorithm with User Queries that can be added; there will be a probability \(\frac{\text{Number of User Queries}}{\text{Total Number of Time Slots}}\) at each time slot, of generating a user query. In parallel we also evaluate our Probabilistic Sampling Algorithm. Since we use probabilities to generate events, it would be computationally too expensive to run all the possible scenarios. Instead we ran these algorithms ten thousand times because we noticed it is enough for them to converge to a certain value (MAE, RMSE and so on).

In the following evaluations, we collect data samples and at the end use interpolation to reconstruct the real data. The interpolation used is third degree interpolation. Third degree interpolation gives more accurate results in exchange for extra computation time. We consider this to not be a problem since all the interpolation calculations should be performed at the gateway who has more processing power than the sensors.

\(^1\)http://sensorScope.epfl.ch/index.php/SensorScope_Deployments
5.2 MAE Comparison

MAE stands for *Mean Absolute Error*. It is a statistical term that we will use for quantifying how much the interpolations using Periodic Sampling and Probabilistic Sampling deviate from the real data. A low MAE is then desirable, since it implies a higher accuracy of the reconstructed data.

Its definition is:

\[
MAE = \frac{\sum_{t=1}^{N}|r_t - i_t|}{N}
\]

Where \(r_t\) is the real data at time \(t\) and \(i_t\) is the prediction; in our case the interpolated point. \(N\) is the total number of data samples that can be collected during a whole day.

In figure 5.1 we examine how the MAE is affected by the choice of the interval size and the number of user queries made to the system. A good result that we should notice beforehand is that the MAE is quite low for both methods.

![Figure 5.1: MAE comparison with varying Interval Size](image)

From the plot we can see that with zero user queries the Probabilistic Sampling method has a lower MAE than the Periodic Sampling method; even though the Probabilistic Sampling method collects more points, they are not as well distributed across the day, as they are with Periodic Sampling.
Once user queries are made, their positive effect is stronger on the Periodic Sampling, which then outperforms our method. Indeed from 0 to 30 user queries, the MAE of Probabilistic Sampling changes very little. This is due to the fact that in our algorithm less auto generated data reports will be made if the number of user queries increases, thus the total number of collected points does not increase as much as in Periodic Sampling.

It should be also noted that the higher the interval size the less data reports are made, thus it is normal the MAE increases with a higher interval size.

Now in figure 5.2 we change the x-axis to a varying number of user queries and try out two interval sizes to see what is the behavior of both methods.

![Figure 5.2: MAE comparison with varying User Queries](image.png)

Using figure 5.2 we verify the conclusions drawn from figure 5.1. Indeed, a lower interval size means more data reports, thus for both methods the MAE is lower for an interval size of 3. One can see, that once the user queries increase, the MAE decreases but it decreases faster for Periodic Sampling because of the reason explained previously.

### 5.3 RMSE Comparison

RMSE stands for Root Mean Square Error. Like the MAE, it gives us a notion of the difference between the real and the interpolated data. The difference though, between MAE and RMSE, is that due to the quadrature,
this statistical term amplifies larger errors, making it more sensitive to them. If there are very large errors, then the RMSE will greatly increase.

Its definition is:

\[ RMSE = \sqrt{\frac{\sum_{t=1}^{N} |r_t - i_t|^2}{N}} \]

As with MAE, \( r_t \) is the real data at time \( t \) and \( i_t \) is interpolated point.

Figure 5.3 shows a very similar behavior to figure 5.1. A change of interval size and number of user queries does not affect the RMSE in any negative way. Due to the definition of the RMSE, it increases with large errors, this explains the larger values on y-axis.

The same applies for figure 5.4, and we can draw similar conclusions as for the MAE analysis; a higher number of user queries and intervals of smaller size are beneficial for the decreasing the RMSE.
5.4 Mean Number of Transmissions

In this section we will compare the mean number of transmissions for both methods of collecting data.

5.4.1 Periodic Sampling

For figure 5.5 we set the User Queries parameter to 15. Its analysis is quite obvious; an increase of interval size decreases the number of reports and the number of sensor replies to queries is always equal to the number of queries from the gateway, as it was expected (but still good to check). We have differentiated the sensor reports that are an answer to a query from the ones a sensor generates by itself.

As for figure 5.6, we set the interval to 5. We can observe that the number of autonomously generated sensor reports remain the same when the interval is set to one value. The number of queries from the gateway increases accordingly to the values of the x-axis.
5.4.2 Probabilistic Sampling

For the Probabilistic Sampling we adopt two views; the communications on the gateway and on the sensor side.
Figures 5.7 and 5.8 concern the gateway transmissions, while Figures 5.9 and 5.10 concern the sensor transmissions.

On the gateway we distinguish two types of transmissions; User Queries are the queries generated by a user and Dummy Queries are queries that the gateway generates without user intervention.

In figure 5.7 we set User Queries to 15; this proportion of transmissions remains constant while Dummy Queries decreases with the interval size.

![Gateway transmissions with varying Interval Size using Probabilistic Sampling](image)

Figure 5.7: Gateway transmissions with varying Interval Size using Probabilistic Sampling

In figure 5.8 we set Interval Size to 5. Dummy Queries decrease slowly when the number of User Queries increases from left to right. When you look at the figure as a whole the total number of transmissions does not change drastically.
Regarding the sensor side we also have two types of transmissions. Sensor Replies to Queries are replies to gateway queries; both user and gateway generated included. Sensor Dummies are data reports sent to the gateway which are initiated by the sensor on its own.

As for figure 5.7 we set User Queries to 15 in figure 5.9. As the interval increases, the number of sensor transmissions decrease as it is the case for the gateway.

In Figure 5.10 we set Interval Size to 5. We can observe that the more a sensor replies to gateway queries, the less it generate reports on its own. In overall the total number of transmissions does not change much with more user queries. This observation backs us up for proving the veracity of the explanation given for the little MAE variation on figure 5.1 once the number of user queries is increased.

Another positive result can be observed from figure 5.10. When the number of user queries is zero, the number of sensor transmissions is 135 in Periodic Sampling while in Probabilistic Sampling it is around 180; thus there are 33\% more transmissions in Probabilistic Sampling than in Periodic Sampling. The interesting part is that when the number of user queries increase, the Probabilistic Sampling maintains a quite constant number of transmissions while it isn’t the case for Periodic Sampling. We can then say,
that when the number of user queries is high, our proposed method saves more sensor energy than the Periodic Sampling.
5.5 Variance of the Number of Transmissions

This section is dedicated to analyzing the variance of the number of transmissions for both methods.

5.5.1 Periodic Sampling

On figures 5.11 and 5.12 the variance of reports generated autonomously by the sensor is equal to zero, this is due to the way the Periodic Sampling algorithm proceeds; the number of sensor reports is equal to the total number of data samples available divided by the interval size. So basically, it always remains the same.

We can observe in figures 5.11 and 5.12 that the variance of transmissions increases with a higher interval size and a higher intervention from the user. Of course the variance of the queries and the variance of its replies is equal.

![Figure 5.11: Variance of Transmissions with varying Interval Size, using Periodic Sampling](image)

Figure 5.11: Variance of Transmissions with varying Interval Size, using Periodic Sampling
5.5.2 Probabilistic Sampling

For Probabilistic Sampling we once again consider the gateway and sensor sides separately. The notation of the plots is the same used as the one in the analysis of the mean number of transmissions, which is found in the previous section.

Figure 5.13 tells us that a higher interval decreases the variance of gateway transmissions due to a smaller number of data requests. Gateway Queries stands for queries originated by the user; in this case we set it to 15, so its variance remains pretty constant, it is the gateway "dummy" generated queries that actually decrease.

In figure 5.14 we set the interval size to 5. If we observe carefully, the variance of gateway generated queries remain stable, it is the higher number of user queries that increase the overall variance of transmissions.

Now we take a look at the sensor’s side. As in figure 5.13, when the interval grows then the variance decreases. Though in this case (figure 5.15) the general level of variance is higher since the sensor has to reply to all gateway queries.
Regarding figure 5.16 where we set the interval to 5, we can see that the variance remains quite stable. If we take a look again at figure 5.10 we see that on average the number of transmissions remains similar even though the number of user queries vary and so the variance results follow accordingly.
Figure 5.15: Variance of Sensor Transmissions with varying Interval Size

It is a good result, since it shows that the variance of communications varies little due to user activity, thus reducing the suspicions of an attacker. Had the variance changed drastically, then an attacker could be sure that some user activity happened during the day when the variance was calculated.

Figure 5.16: Variance of Sensor Transmissions with varying User Queries
5.6 Attacker’s Methodology

To test how resistant the system is to a traffic analysis by an attacker, we simulated an attack.

Supposing the attacker knows what interval size the system is using, he can compute what is the average number of gateway queries during an interval of time. Then, the attacker goes through all the intervals and checks if the number of gateways queries in that interval differs from the previously computed average, by a certain threshold $\alpha$. If the difference is higher than the set threshold, then the interval is considered suspicious by the attacker, meaning that probably there was a user query during that interval.

5.6.1 Attacker’s Algorithm

Here below is a pseudo code for the attacker’s algorithm for determining intervals who contain user activity. Note that ”#” stands here for number of; this notation was adopted to shorten the writing of the algorithm.

\begin{algorithm}
\caption{Attacker’s Algorithm}
\begin{algorithmic}[1]
\Require \textit{Interval Size, Threshold $\alpha$}
\State \textbf{for} each Interval \textbf{do}
\State Count #(Gateway Queries)
\EndFor
\State Calculate Average #(Gateway Queries) per Interval
\State \textbf{for} each Interval \textbf{do}
\State \textbf{if }$|\#(\text{Gateway Queries}) - \text{Average } #(\text{Gateway Queries})| > \alpha$ \textbf{then}
\State Mark Interval as having User Activity
\State \textbf{end if}
\EndFor
\end{algorithmic}
\end{algorithm}

As indicated before, the parameters for the algorithm are the interval size and the threshold value $\alpha$.

5.6.2 Attacker’s Success Rate

To characterize how successful an attacker can be in determining we define the attacker’s success rate in the following way:

\[
\text{Attacker’s Success Rate} = \frac{\text{Number of Suspicious Intervals}}{\text{Number of User Queries + Number of Wrong Guesses}}
\]
As one can observe from the figure 5.17, the choice of the threshold by the attacker is not an easy choice, some threshold values give a higher success rate than others. A smaller threshold value does not necessarily mean a higher success rate of the attacker, indeed the threshold value of 0.25 gives a lower success rate than the threshold value of 0.5. The reason for this will be explained in the next section about False Positives and False Negatives.

In figure 5.18 we set the interval size to 5 and observe how the success rate of the attacker depends on the number of user queries made to the system.

We can see that the graph is very similar to the one in figure 5.17. Though a choice of threshold value does not offer a linear behavior of the attacker’s success rate, we can still give two recommendations for a better security.

The user of this kind of system should use a small interval size and as long as he does not query too much the system, then he will have good privacy. This does not seem unreasonable; few users would make 100 queries in a day to check the temperature of their room for example.
5.6.3 False Positives & False Negatives Rates

After analyzing the attacker’s success rate, we will take a look at the rate of intervals that are False Positives and False negatives.

The rate of False Positives is formally defined as:

\[ \text{False Positives} = \frac{\text{Number of Wrong Guesses}}{\text{Number of Suspicious Intervals}} \]

Similarly the rate of False Negatives is defined as:

\[ \text{False Negatives} = \frac{\text{Number of User Queries Undetected}}{\text{Number of User Queries}} \]

An interval is a False Positive if the number of gateway queries differ from the average by the defined threshold, but still does not contain a user query. In this case the attacker is mislead to think that interval contains a user query though it is not the case. False positives will increase the wrong information that an attacker can possess about the user’s patterns of activity.

An interval is a False Negative if it’s average number of queries does not differ from the average by the set threshold but did in fact contain a user query. These types of intervals are disregarded by the attacker who does not suspect them of containing user queries.
In figure 5.19, the threshold value was set to 1.5, on the y-axis we put the number of mistakes an attacker makes and we observe out of those mistakes the proportion of false positives and false negatives. In this figure we can also see that the number of false negatives remains quite constant and the number of false positives decreases with a higher interval size. Which means, the larger the interval size, the higher the attacker’s success rate.

![Figure 5.19: Attacker’s mistakes with varying Interval Size and User Queries = 15](image)

What is really interesting in this figure is that the number of false negatives is much higher than the one of false positives.

What this means is that with a threshold value of 1.5 we are too far away from the average number of gateway queries per interval, thus we are excluding too many intervals of suspicion. In contrast if we picked a threshold value of 0.25 (as can be seen in figure 5.17), then at some point we will have more false positives than false negatives; the attacker will suspect too many intervals.
5.7 Remarks

In these evaluations we always used a zero probability for generating unusual events, the reason for this is that this probability is supposed to be quite low like the name suggest, so its effect is negligible. Since the sensors generate autonomous data reports, it will be very unlikely for the attacker to be able to identify a slot of time during which an unusual event happened. Indeed, it depends on the definition of what an unusual event is, but we suppose that it should not occur several times a day.

I would like to add, that we supposed the attacker knew the interval size adopted by the system. We did it in order to simplify the execution time complexity of the attack performed on our system, but in reality the attacker would eventually need to try out different interval sizes, doing an extensive number of computations. So the time for an attacker to be able to identify the intervals with user activity is the time necessary to run the attacker’s algorithm presented previously, times the number of possible interval sizes. It can also happen that the attacker considers the wrong interval size when analyzing the traffic, which makes our method even more resistant. Also to take in account is the choice of the parameter $\alpha$ which is not obvious, the attacker needs to try a few values and still he will never reach 100% certainty in its guesses.

Finally the attacker’s method relies on an average that is only calculated once an entire day has passed, meaning that only at the end of the day the attacker can analyze user activity. Another possibility for the attacker would be to compute an average progressively after each interval of time.
Chapter 6

Conclusion & Future Work

It is very difficult to completely mask user activity due to its unpredictability. Even though it does not hide completely the user’s activity, the Probabilistic Sampling algorithm still brings some security to the system.

There is much room for improvement while taking some trade-offs into account. We could for example put a higher transmission probability on the gateway side in order to better hide user queries, but we would elevate the cost of transmissions. We could also try to minimize even further the number of communications, but we would be decreasing the level of security.

Another important aspect is the security evaluation of our algorithm. It is hard to come with appropriate security metrics [31] and we could have proposed a more complex attacker’s algorithm, but I think that could be another thesis topic.

It should be noted that the Probabilist Sampling method was proposed to work in a scenario as described in figure 3.1. Thus I think we use a pretty simple protocol for organising transmissions between the gateway and the sensors. If you notice, there is no need to store nor exchange a transmission schedule between parties.

Finally, our approach can show beneficial side effects. An extended usage of our proposed system is to combine readings from sensors within a particular area; if one sensor stops working, then two nearby sensors can be used to calculate an average of their readings in order to estimate the values of the non-working sensor. Since the two nearby sensors report data at different times, by combining them, the accuracy of the interpolation is higher since we will have a better sampling coverage of a day.
Figure 6.1: Better day coverage by combining A and B sensor readings
Bibliography


[9] Durvy, Mathilde and Abeillé, Julien and Wetterwald, Patrick and O’Flynn, Colin and Leverett, Blake and Gnoske, Eric and Vidales, Michael and Mulligan, Geoff and Tsiftes, Nicolas and Finne, Niclas


