ARTIFICIAL SATELLITES, Vol. 48, No. 2 – 2013
DOI: 10.2478/arsa-2013-0006

DETECTING MEACONING ATTACKS BY ANALYSING THE CLOCK BIAS OF GNSS RECEIVERS

Daniel Marnach[1], Sjouke Mauw[2], Miguel Martins[1], Carlo Harpes[1]
itrust consulting s.à r.l. & University of Luxembourg
[1]: {surname}@itrust.lu, [2]: {name.surname}@uni.lu

ABSTRACT

Existing Global Navigation Satellite Systems offer no authentication of their satellite signals towards their civilian users. As a consequence, several types of GNSS-related attacks, including meaconing, may be performed and remain undetected. In the scope of the project *"Developing a prototype of Localisation Assurance Service Provider"*, which is funded by ESA and realised by the company itrust consulting and the University of Luxembourg, a methodology to visualise the beginnings and the ends of meaconing attacks by monitoring the clock bias of an attacked receiver over time was developed. This paper presents an algorithm that is based on this attack visualisation technique and is capable of detecting meaconing attacks automatically. Experiments in a controlled environment confirmed that the chosen methodology works properly. In one of these tests, for example, six meaconing attacks were simulated by using a GNSS signal repeater. The algorithm was able to detect the beginnings and the ends of all six attacks, while resulting in no more than two false positives, even though the average delay introduced by the meaconing stations (repeater) was just 80 nanoseconds.

Paper presented at the"European Navigation Conference 2012",
held in Gdansk, Poland

1. INTRODUCTION

Over the last two decades, the use of Global Navigation Satellite Systems (GNSS) has become extremely popular. Thanks to the freely available positioning systems GPS, the Russian alternative GLONASS, and in a couple of years also the European constellation Galileo, navigation systems are not only available to the military but also accessible in the civil domain with a relatively high precision. In addition, the decreasing price of GNSS receivers as well as the trend of using more and more mobile devices led to the integration of the use of positioning systems in most people's everyday life.

In the meantime, an important number of Location-Based Services (LBS) became available. A Location-Based Service Provider (LBSP) offers a service to a customer that depends on the customer's location. The customer uses a GNSS enabled device to make a request to the LBSP. This request includes the customer's position, which influences the answer to the request. A typical example is location-based access control, where access to a resource is granted only if the access request was issued from within a certain area. Other examples include the tracking of valuable assets or hazardous materials by a transport management centre, the enforcement of road toll payment by governments, and the monitoring of journalists in dangerous areas of the world.

As explained in (Scott, 2007), there are good reasons to attack such systems, mainly because of prospects for financial gain. Moreover, the transmission of information takes place over radio links, which are by their very nature insecure channels (Hein et al., 2007), making it even easier for a potential attacker to succeed. The military and commercial positioning services are encrypted and hence offer sufficient protection against most attacks. The freely available services, however, neither are encrypted nor signed by the satellites and thus guarantee neither the integrity of the GNSS signals nor the authentication of the sender of the signals. In other words, open services offer no built-in functionalities to their users to check whether the received signals really originate from a navigation satellite and whether the signals were altered by a third party since they were broadcast.

As civilian positioning applications will be used more and more in fields related to safety and security (Hein et al., 2007), it is unavoidable to also think about security mechanisms to protect the users of the free positioning services. Several solutions that intend to ease authentication by modifying the current structure of the GNSS signals have been proposed. One could, for example, use a signature scheme, i.e. compute a hash value of the message before sending it, sign this hash with a private key only known to the space segment (satellites) and the control segment (authorised ground stations), and append the signature to the message. A corresponding public key could then be used by anybody to verify the authenticity of the message. This technique is called Navigation Message Authentication (NMA) and is described, for example, in (Hein et al., 2007). However, as explained in the same article, NMA and similar techniques bring their own problems and as stated in (Stansell, 2007), it is unlikely that modifications to the GNSS signal structures will be done in the near future. Therefore, it is important to find location assurance solutions that do not require any changes to the structure of the GPS, GLONASS and Galileo signals. In (Harpes et al., 2009), the authors propose the architecture depicted in Fig. ***1*** as a possible solution.

The idea is the following: After having computed its location from the captured GNSS signals, the user device (UD) sends a set of data to a Location Assurance Provider (LAP). This data includes the computed location, as well as other information that can be derived from captured signals or from the user device itself. The LAP analyses the data and decides whether or not it is authentic. It then returns a certificate to the user device that is bound to the previously received data set and that contains an assurance level. The higher the assurance level, the more trustworthy is the data set according to the LAP. The user device forwards the received certificate to a Location-Based Service Provider (LBSP) instead of just sending the computed location. Finally, the LBSP can use the public key of the LAP to check whether the certificate is authentic. In addition, a Public Key Infrastructure (PKI) is used to check whether the certificate was really issued by the intended LAP.

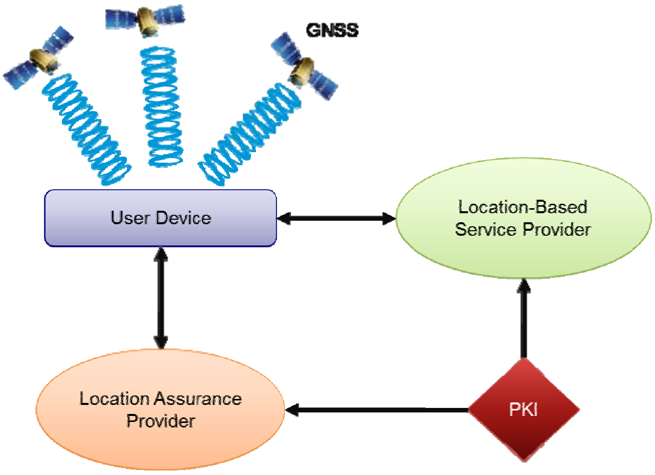

Fig. 1. The LASP architecture – inspired by (Harpes et al., 2009)

In their current project "*Developing a prototype of Localisation Assurance Service Provider (LASP)*", itrust consulting s.à r.l. and the University of Luxembourg aim at implementing the architecture presented above, which is in the following referred to as LASP architecture. The main security requirements of the project are given below:

- Let L be a location. Then, a user either is or is not located within a certain radius R around L. If he claims towards an LBSP to be at L, the confidence that he is indeed at a maximal distance R from L is at least as high as implied by the assurance level issued by the LAP.
- If a user wants to connect to the LAP or a LBSP, he should be able to do so.
- Users should be able to control the amount of personal data that is sent to service providers (e.g. the precision of their location).

From these requirements, we identified four security properties, namely confidentiality, integrity, availability and authenticity. In this document, however, we focus on meaconing attacks only. Meaconing is the interception and rebroadcast of navigation signals in order to confuse navigation (Harpes et al., 2009). It does not modify the signals but delays their arrival at the GNSS receiver. Since the sender of the relayed signals is not a satellite, the source of the messages is not the intended one and hence, meaconing attacks break the authenticity property.

Since navigation satellite systems are based on the concept of time of arrival ranging (cf. Section 1.2), the introduced delay falsifies the computed user positions. The data sets that contain (among others) the false information are sent to the LAP, whose task it is to detect this and similar types of frauds. To do so, it runs in a first step a number of so-called security checks. These are algorithms that get as input specific parts of the received data sets and output their opinion on whether the analysed information is authentic. Thereby, the output is for most checks not a Boolean answer, but a confidence degree, i.e. a real number between 0 and 1 expressing the plausibility of the claimed location (A confidence degree of 1 means that in the opinion of the security check, the claimed location is authentic for sure). In a second step, the outputs of all executed security checks are merged in order to determine the assurance level.

In this paper, we present the clock bias security check, an algorithm that was explicitly designed for the detection of meaconing attacks. Due to the fact that these attacks always involve signal delays, timing information can be considered to recognise the attacks' beginnings and endings. The idea of performing time comparisons in order to detect GNSS-related attacks was already mentioned in some publications (e.g. (Scott, 2007)), but to the best of our knowledge, we are the first to put it into action and thus to face issues related to its implementation.

The remaining part of this document is structured as follows: Section 2 introduces the theoretical background that the clock bias security check is based on and Section 2 discusses the experimental setup that was used for the development of the check. Finally, the algorithm itself is presented in Section 3, followed by a summary of the paper and a discussion on future work in Section 0.

2. THEORETICAL BACKGROUND

In this section, we provide the background knowledge that the reader needs to understand the basic idea behind the clock bias security check. Except where indicated otherwise, the information provided in Section 0 is based on (Kaplan et al., 2005).

1.1. TIMING INFORMATION

Each satellite and each GNSS receiver is equipped with a clock, and the difference in time between such a clock's value and a pre-defined reference time is referred to as clock bias (Hurn, 1989). In contrast, the drift of a clock is expressed in seconds per second (s/s) (El-Rabbany, 2006). It is the first derivative of the clock bias with respect to time. Hence, a clock bias arises due to the fact that clocks drift over time.

The smaller the drift of a clock, the better (the closer to zero) is its accuracy. Satellites, on the one hand, have an atomic clock on board (El-Rabbany, 2006). This type of clock offers a very high accuracy (e.g. 10^{-14} (Novick, 1994)) and is therefore nearly drift-free. Thus, we consider the clock bias of a satellite clock towards a non-drifting reference to be constant. Ground stations that monitor the satellites are equipped with atomic clocks, too (El-Rabbany, 2006). Receivers, on the other hand, are usually equipped with a crystal clock to minimise the cost and the size of the user devices. Crystal clocks are by nature far less accurate than atomic clocks. In addition, their drift is influenced by environmental conditions, like temperature (Vig, 2008).

A clock may be synchronised with its reference time. In other words, the clock's value may be adjusted to avoid letting its clock bias towards the reference become too large. We call synchronisation instant an instant of time at which synchronisation occurs.

A number of time reference systems are used worldwide, two of which are UTC (Coordinated Universal Time) and GPS time (El-Rabbany, 2006). The former is related to the rotation of the earth and must occasionally be adjusted to keep it synchronised with the planet's solar time. This is done by adding so-called leap seconds. In contrast, GPS time is computed from the time scales generated by the atomic clocks of both the GPS satellites and the GPS ground segment. It is a continuous time scale, i.e. no leap seconds are applied. Originally (in 1980), GPS time was consistent with UTC. However, the latter was increased 15 times by one second since then and therefore is 15 seconds ahead of GPS time, now. Since GPS is currently the most important Global Navigation Satellite System, GPS time is used as reference time, here. Hence, we define the clock bias *b* of a receiver at time *t* to be the difference between the receiver's clock value τ_r at time *t* and GPS time τ_{GPS} at time *t*:

$$b(t) = \tau_r(t) - \tau_{GPS}(t) \quad [s].$$

As it is common in computer science, we consider time to be a discrete quantity and therefore choose a vector representation for physical values that are not constant over time. In other words, a function *f(t)* with $0 \le t < n$ (and *t* an integer) is represented by the vector $(f_0, \ldots, f_{n-1})$. Thereby, f_t is equal to *f* evaluated at time *t*. For example, b_t denotes the clock bias at time *t*.

1.2. POSITION DETERMINATION

A GNSS receiver captures navigation signals broadcast by satellites to determine its position at time *t*. The coordinate system that is used to compute such positions rotates with the earth and its origin is located at the centre of the planet. Therefore, it is referred to as Earth-Centred Earth-Fixed (ECEF) system (Borre et al., 2007). The x-axis points towards the intersection between the equator and the Greenwich meridian and the z-axis points towards the geographical North Pole, i.e. it overlaps with the spin axis of the earth. The y-axis is orthogonal to the x- and the z-axis and thereby forms a right-handed coordinate system. Note that the x- and the y-axis together define the equatorial plane.

A vector representation of the receiver's position in the ECEF system at time *t* is given in Fig. *2*. The position of the receiver is denoted by r_t, while that of satellite *i* is denoted by $s_t^{(i)}$. Furthermore, $u_t^{(i)}$ refers to a vector that has its origin at the receiver and points towards satellite *i*. Its length $|u_t^{(i)}|$ is equal to the Euclidean distance between the receiver and the satellite in meters.

Thus, determining the receiver's position at time *t* is equivalent to computing the vector r_t. In the following, a solution to this problem is discussed. The time indicating indices are omitted, as the position determination procedure given below does not rely on information related to a time $t' \neq t$ to calculate a receiver's position at time *t*.

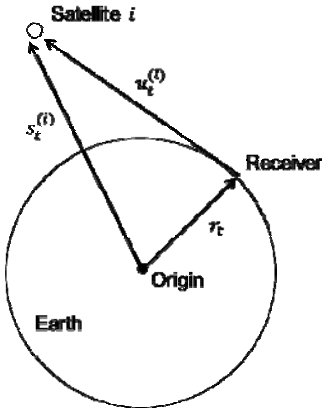

Fig. 2. Vector representation of the receiver position at time t – inspired by (Kaplan et al., 2005)

The idea of the position determination procedure is to compute *r* from $s^{(i)}$ and $u^{(i)}$. Thereby, $s^{(i)}$ can be computed by the receiver based on data contained in the navigation signals broadcast by *i*. In contrast, $u^{(i)}$ cannot be found directly. The subsequent paragraphs explain how it may nevertheless be used to compute *r*.

Global Navigation Satellite Systems are based on the concept of time of arrival ranging. When a signal arrives at the receiver, the value of the receiver's clock is compared to timing information contained in the signal to measure the distance between the receiver and the satellite that broadcast the signal. If the receiver's clock was perfectly synchronised with the satellite's clock, the measured value $\rho^{(i)}$ would be equal to the physical distance $|u^{(i)}|$ between the objects. As explained in Section 1.1, however, only the satellite's clock is considered to be drift-free and hence the clock bias b of the receiver cannot be assumed to be zero. As a consequence, $\rho^{(i)}$ is biased by the term $c \cdot b$, where c is the speed of light (299792458 m/s), i.e. the traveling speed of the signal:

$$\rho^{(i)} = \ |u^{(i)}| + \ c \cdot b \ [m].$$

So, $\rho^{(i)}$ is in general different from $|u^{(i)}|$ and is therefore called the pseudo-distance (or pseudo-range) between the receiver and satellite i.

As $\rho^{(i)} = |u^{(i)}| + c \cdot b$ and $u^{(i)} = s^{(i)} - r$, it holds that $\rho^{(i)} = |s^{(i)} - r| + c \cdot b$. In a 3-dimensional setting, r is equal to (r_x, r_y, r_z), which means that including b, there are four unknowns. Thus, a minimum of four equations is required to solve the problem, which implies that at least four satellites must be visible to determine a receiver's position. As a side effect, solving the equations also provides the receiver's clock bias b, which is essential for the clock bias security check.

2. SYSTEM DESCRIPTION

In this section, we provide details on the experimental setup that was used for the development of the clock bias security check. This includes on the one hand a description of the employed LASP user device and on the other hand a way to simulate meaconing attacks.

2.1. THE LASP USER DEVICE

Typically, one would think about a user device as a single piece of technology such as a smartphone. At the current stage of the project, however, the user device of the LASP architecture is composed of three components, namely a GNSS antenna, a GNSS receiver and a personal computer (Fig.3). The antenna (Novatel GPS-703-GGG) and the receiver (JAVAD Delta-TRE) are professional equipment and therefore offer more functionalities than standard GNSS devices. They are linked via a coaxial cable and the receiver is connected to the personal computer via Ethernet. The reader should be aware that the choice of these components also influenced the design of the security check described in this paper.

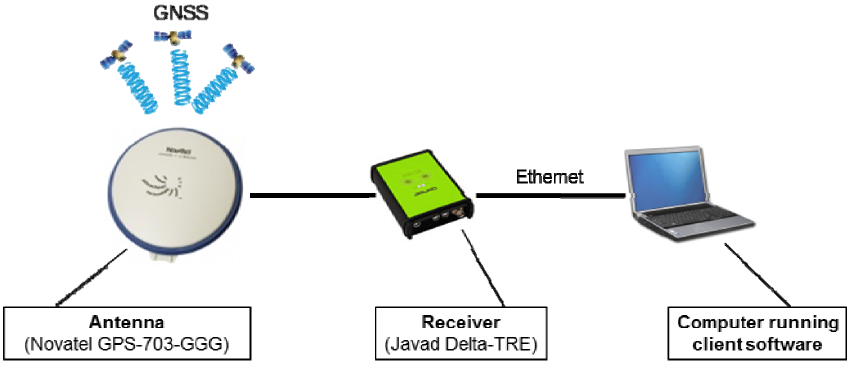

Fig.3. The user device of the LASP architecture

2.2. RECEIVER BEHAVIOUR

As the receiver JAVAD Delta-TRE is part of the user device, its behaviour had to be known in order to develop an efficient security check. The device's functionalities are documented ((JAVAD, 2010) and (JAVAD, 2011)), but not to the level of detail that was required for this work. Therefore, several experiments were conducted in order to reverse engineer the receiver's specific behaviour. It follows a summary of the findings that turned out to be relevant for the design of the clock bias security check. A more detailed presentation of these results can be found in (Marnach, 2012).

A synchronisation mode is a methodology that defines when and how a receiver's clock will be synchronised with a pre-defined reference time. JAVAD Delta-TRE offers only one synchronisation mode that is suitable for the type of security check described in this document. It is called *ms* and defined as follows in (JAVAD, 2011): when the time difference (clock bias) exceeds 0.5 milliseconds, correct receiver time by 1 millisecond. This synchronisation mode was active during all our recordings.

We recall that the clock bias b of a receiver is the difference between the receiver's clock value and GPS time. A sample recording of our receiver's clock bias over time is depicted on the left side of Fig. *4*. The plotted curve is called clock bias function. As the receiver's synchronisation mode is set to *ms*, the absolute value of the clock bias is allowed to grow over time. In this case, the receiver clock runs faster than its reference. As a consequence, one millisecond is subtracted from the clock bias each time it becomes larger than half a millisecond. If this happens at time t, then t is a synchronisation instant and $t-1$ is a pre-synchronisation instant, i.e. the last instant of time before synchronisation occurred. A part of the clock bias function that starts at a synchronisation instant and ends at a pre-synchronisation instant is called a complete period. It is important to note that different periods have in general different lengths and that each period is a set of (t,b)-points, where t is a time value and b the receiver's clock bias at that time. The clock bias function is hence a time-discrete function. Moreover, it was found out that GNSS signal outages have no influence on the clock bias, except interrupting the curve for the duration of the loss.

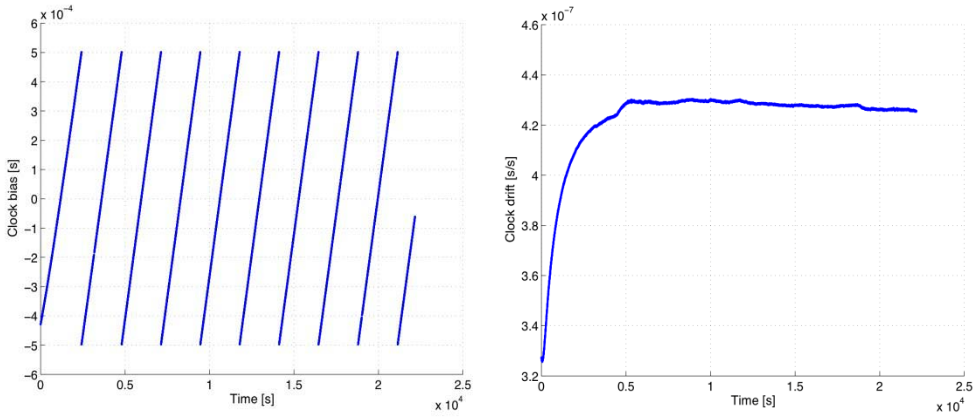

Fig. 4. Clock bias (left) and clock drift (right) over time

Since different periods have different lengths, one can conclude that the time derivative of the clock bias is not constant. We recall that this derivative is equal to the clock's drift, which is expressed in seconds per second. The clock drift function corresponding to the clock bias function discussed earlier is shown on the right side of Fig. *4*. As expected, it is not constant over time and hence, periods are not straight lines. Also, it was found out that the clock drift is strongly influenced by the receiver's temperature. In fact, the relation between the drift and the temperature is almost linear, which means that other environmental influences on the drift may be neglected. After switching on the receiver (at room temperature), it takes about 100 minutes for the receiver temperature to become stable.

2.3. MEACONING ATTACKS

For now, we do not have the technical means to perform real meaconing attacks. Instead, we simulate such attacks on the LASP user device by employing a GPS signal repeater.

A repeater receives GNSS signals and forwards them without modifying their content. The signal processing will, however, take an amount of time $\Delta t > 0$ to be completed. Thus, a repeater always delays the signals it forwards. In Section 1, it was explained that a meaconing attack does not modify signals but delays their arrival at receivers. Hence, the effect of a meaconing attack and the effect of a repeater usage on GNSS signals are comparable.

The repeater (which comes with its own antenna) and the antenna of the LASP user device are both connected to the inputs of a coaxial switch (Fig. *5*). The latter's output is then connected to the signal input of the receiver. In this way, it is possible to switch between two user device architectures, one containing the repeater and one not containing it. Thereby, the architecture that contains the repeater symbolises a device that is under attack and hence, switching back and forth between the architectures once represents the beginning and the end of an attack.

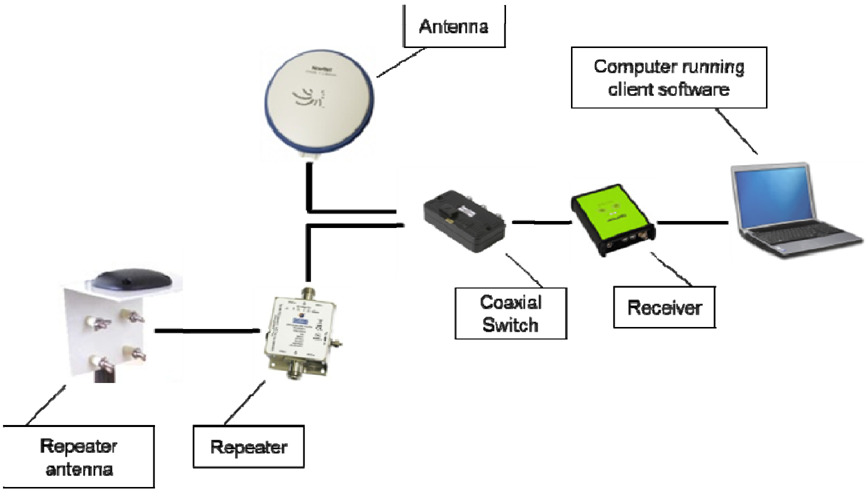

Fig. 5. Adding a repeater to the LASP user device

3. THE CLOCK BIAS SECURITY CHECK

This section covers the development of the clock bias security check. First, the mathematical theory behind the check is introduced. It is based on the theoretical background provided in Section 0. Second, the ten steps of the algorithm are discussed one by one.

3.1. BASIC IDEA

We suppose that it is possible to detect meaconing attacks by observing the clock bias of a GNSS receiver over time. In this subsection, it is explained why it is realistic to make such an assumption and how the influence of an attack on the clock bias is assumed to look like.

From Section 1.2 it is known that at time t, $\rho_t^{(i)} = |u_t^{(i)}| + c \cdot b_t$, where i is a satellite, $\rho_t^{(i)}$ the pseudo-distance between the receiver and i, $|u_t^{(i)}|$ the real distance between both objects, c the speed of light and b_t the receiver's clock bias.

Assume now that a meaconing attack M, starting at time g and ending at time h (i.e. the domain is $[g,h]$), is performed to delay the arrival of i's signals at the receiver. This directly affects the pseudo-distance since the latter's measurement is based on the signals' travelling time. As $\rho_t^{(i)}$ is equal to $|u_t^{(i)}| + c \cdot b_t$, it holds that $b_t = (\rho_t^{(i)} - |u_t^{(i)}|) / c$. In a first step, suppose that the location of the receiver and the satellites would remain static. With $\rho_{\mathrm{g}}^{(\mathrm{i})} > \rho_{\mathrm{g}-1}^{(\mathrm{i})}$ and $\rho_{\mathrm{h}+1}^{(\mathrm{i})} < \rho_{\mathrm{h}}^{(\mathrm{i})}$, while $|u^{(i)}|$ and c both remain constant over time, it is clear that $b_g > b_{g-1}$ and $b_{h+1} < b_h$. So, by delaying signals from satellite i, an adversary artificially modifies a receiver's clock bias, if the receiver uses i's signals to compute its position. We conclude that it should be possible to detect the beginning and the end of M by looking for changes of the clock bias over time. To be precise, an increase of the clock bias refers to an attack's beginning, while a decrease alludes to its end. The larger the signal delay caused by M, the bigger the difference between b_g and b_{g-1}, and between b_h and b_{h+1}. In a second step, it must be considered that the clock's natural drift influences the clock bias, too, and that the physical distance between the receiver and the satellites does not remain constant over time. However, we expect these influences to not result in unpredictable abrupt changes of the clock bias.

An adversary who tries to falsify computed positions by using meaconing attacks has to introduce a large delay in order to significantly deviate from the receiver's real position. This can either be done by applying one large delay or by starting with a very small delay which is then increased over time. The first method results in one abrupt change of the clock bias, while the second one results in a sequence of changes that can also be considered as abrupt unless the gradual changes of the delay are really small. If we assume that it is not feasible to effectively falsify a position by gradually changing the clock bias, it should be possible to detect meaconing attacks by looking for abrupt changes of the clock bias function.

3.2. ATTACK VISUALISATION

As an experiment, our receiver's clock bias was recorded while the LASP user device was under a meaconing attack. This was realised by adding a GNSS signal repeater to the user device architecture (cf. Section 2.3) after about 5770 seconds of recording and by removing it again after about 6440 seconds of recording. So, the data collection started prior to the beginning of the attack and terminated after the end of the attack.

The goal of the test was to manually detect the beginning and the end of the attack by applying the theory described in the previous section, i.e. by looking for abrupt changes of the clock bias over time. However, it turned out that the abrupt changes of the clock bias function are not visible to the naked eye. Thus, another graphical representation of the recorded values was chosen. It is specified in the subsequent paragraph.

Let S be a set of (t,b)-points. Then first, a regression line with respect to S, i.e. a line that "fits best" all the points in S, is computed. Thereby, $b^{(S)}(t) = \alpha^{(S)} \cdot t + \beta^{(S)}$ denotes the regression line's equation and the method of least squares is applied to determine the coefficients $\alpha^{(S)}$ and $\beta^{(S)}$. Second, we define the clock bias error e at time t with respect to S to be the difference between the clock bias b at time t and the evaluation of $b^{(S)}$ at time t:

$$e_t^{(S)} = b_t - b^{(S)}(t)\,.$$

The clock bias error is of much smaller magnitude than the clock bias itself and therefore, it is possible to make the effects of the meaconing attack visible by plotting the clock bias error over time. For this particular experiment, S was chosen to be the set of all (t,b)-points such that $t \in [4822, 7160]$, which corresponds to the period during which the attack occurred. The resulting clock bias error function $e^{(Period)}$ is depicted in Fig. ***6***. There are two abrupt changes of this function, one at $t = 5778$ and one at about $t = 6440$. They symbolise the beginning and the end of the attack, respectively.

It was decided to base the new graphical representation of the recorded data on linear regression because the observed clock bias function (Fig. ***4***) is approximately linear. However, it could be that for other types of receivers, approximation polynomials of higher degree are preferable.

The experiment showed that it is indeed possible to visualise a meaconing attack and hence, it should also be possible to develop an algorithm that is able to detect such attacks automatically.

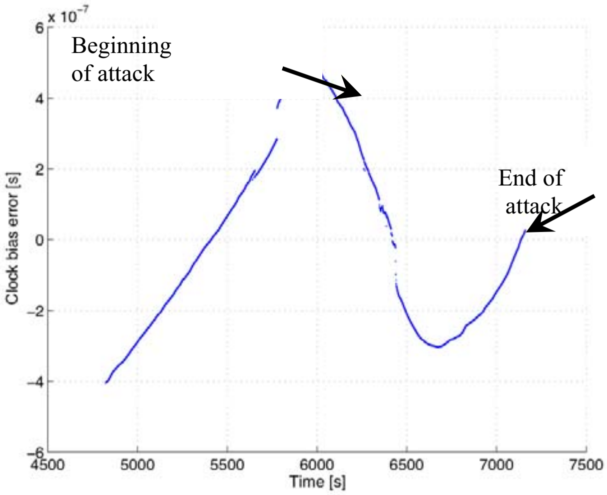

Fig. 6. Clock bias error

3.3. DETECTION ALGORITHM

In this section, an algorithm that is capable of detecting the beginning and the end of meaconing attacks automatically is presented. Its design is strongly inspired by the attack visualisation technique described previously.

3.3.1. ASSUMPTIONS

The following assumptions are made for the development of the detection algorithm:

- The data interval is the time in seconds that elapses between the recording of two consecutive clock bias values at the user device. It is not identical for all types of devices and may also be changed over time for a particular unit. For now, however, we assume that data intervals are constant over time.
- Each data set that is passed to the Location Assurance Provider by the user device contains at most one reference time value and one clock bias value. It is supposed that the Location Assurance Provider is capable of extracting these values from the set in order to form (t,b)-points.
- It is assumed that the receiver temperature is more or less constant over time. In principle, this means that the device has been running for at least 100 minutes (cf. Section 2.2).

3.3.2. LEAPS

Henceforth, we refer to an abrupt change of the clock bias error function by the term leap. Thereby, a leap is not a scalar, but a set of (t,e)-points. In the following paragraphs, the vocabulary that is used to reason about leaps is specified.

Let λ be a leap. Then λ visually divides a plotted clock bias error into two parts, which we call the pre-leap curve A of λ and the post-leap curve B of λ (Fig. 7).

Let g denote the end time of A and h the starting time of B. Then (g,e_g) and (h,e_h) are the last point of A and the first point of B, respectively, and we say that λ starts at g and ends at h. As a consequence, the difference between h and g is called the leap duration of λ: $duration(\lambda) = h - g$.

There may be some (t,e)-points such that $g < t < h$, i.e. that belong neither to A, nor to B. In that case, we say that λ includes detached points. The leap depicted in Fig. 7, for example, includes one such point. Due to detached points, the duration of a leap is not always equal to the data interval.

We define the leap value of λ to be the difference between e_h and e_g. Moreover, we refer to the absolute value of the leap value as leap height: $value(\lambda) = e_h - e_g$ and $height(\lambda) = |value(\lambda)|$.

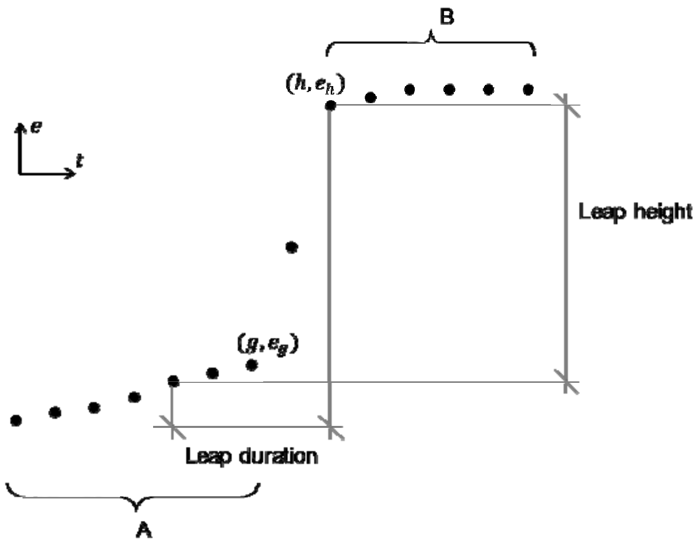

Fig. 7. A sample leap

The reader should be aware of the fact that the definition of the term leap depends on the visual division of a plotted curve into two parts *A* and *B*. So, the division criterion, which is not well-defined, has an influence on the dimensions of a leap λ. In case of the end of attack depicted in Fig. ***6***, for example, it is hard to tell exactly which two points mark the beginning and the end of the leap caused by the attack. However, if the end time and the duration of λ are known, the value of λ is well-defined: $value(\lambda) = e_h - e_{h-duration(\lambda)}$.

3.3.3. OBSERVATIONS

By simulating further meaconing attacks and analysing the collected data, the following two observations were made:

- A leap ending at time *h* can also be visualised by using merely the (*t,b*)-points such that $t \le h$ to compute the regression line. Thus, it is possible to detect a leap immediately after its triggering event (beginning or end of attack) occurred.
- Among all the leaps that were detected manually, the minimum height was about $0.65 \cdot 10^{-7}$ seconds and the average height was about $0.8 \cdot 10^{-7}$ seconds. This means that the smallest delay introduced by the employed repeater was 65 nanoseconds and the average delay was 80 nanoseconds.

Please note that both observations are based on experiments that were conducted with a data interval of 1 second.

3.3.4. ALGORITHM DESIGN

The detection algorithm is executed once for each (*t,b*)-point that arrives at the Location Assurance Provider. We recall that each data set that is passed to the LAP by the user device of the LASP architecture contains at most one of these points.

Thereby, missing points (sets without timing information) indicate that signal outages occurred. The algorithm's input and output is specified below:

- Input:
  - The arriving (t,b)-point P.
  - n (t,b)-points that arrived in the past, where n is a pre-defined integer larger than zero. Thereby, it is recommended to use the n points that directly preceded P.
  - The data interval of the user device (cf. Section 0).
- Output:
  - The confidence that the time value t of P is not the starting time or the end time of an attack. This is a convention of the LASP project.

The history length, denoted by *histLength*, is defined to be the total number of (t,b)-points that are provided to the algorithm, hence $n + 1$.

Let *time*[] and *bias*[] be two arrays of size *histLength*, containing the time values and the clock bias values of all the (t,b)-points that are provided to the algorithm in chronological order. Thereby, (*time*[0], *bias*[0]) denotes the oldest point and (*time*[n], *bias*[n]) the one that just arrived at the Location Assurance Provider. Then, *time*[n] is automatically suspected for being the end time of a leap λ and the algorithm's task is to give its opinion on whether *time*[n] is "guilty" or "not guilty". To do so, the following ten steps are executed:

(1) Set parameters

The parameters *minLeapDuration*, *detectionBound*, *minConfidence* and *maxConfidence* are set. Their meanings are given below:

- As explained in Section 0, the end time of a leap and the leap's duration must be known in order for the leap's value to be well-defined. Here, λ's end time is equal to *time*[n], but its duration is unknown. Hence, the latter is estimated. The parameter *minLeapDuration* denotes the estimated duration.
- The *detectionBound* is a value to which λ's height is compared to in order to decide whether *time*[n] is guilty or not guilty
- *minConfidence* and *maxConfidence* denote the minimum and the maximum possible values that are output by the algorithm.

(2) Gather information about the window

The set of all (t,b)-points that are provided to the algorithm is called the algorithm's window. We distinguish between the length and the duration of the window, denoted by *winLength* and *winDuration*, respectively. *winLength* represents the number of points that are in the window and is hence equal to *histLength*, while *winDuration* stands for the time span (in seconds) that is covered by the window. It is equal to the difference between *time*[n] and *time*[0]. Please note that

$winDuration \neq (winLength - 1) \cdot dataInterval$ if any signal outages occurred inside the window.

(3) Undo synchronisation inside the window

Leaps are not only caused by the beginnings and the ends of meaconing attacks, but also by clock synchronisations. As a consequence, synchronisation instants would be thought guilty. In addition, synchronisation complicates the computation of regression lines. These problems can, however, be prevented by undoing synchronisation inside the window. If there is an s such that $time[s]$ is a synchronisation instant and $time[0] \leq time[s] \leq time[n]$, then for each x such that $time[s] \leq time[x] \leq time[n]$, the value of $bias[x]$ is modified with the objective of simulating a scenario in which synchronisation did not occur.

(4) Compute a regression line with respect to the window

A regression line $b(t)$ with respect to the algorithm's window is computed. As explained in Section 3.2, that means that the coefficients of the line $b^{(Window)}(t) = \alpha^{(Window)} \cdot t + \beta^{(Window)}$ must be determined.

(5) Determine the leap's starting time

Since λ's duration $duration(\lambda)$ is the difference between the end time h and the starting time g of λ, it holds that $g = h - duration(\lambda)$. Here, h is equal to $time[n]$ and $duration(\lambda)$ is given by the parameter *minLeapDuration*. However, this parameter is an estimated value and hence, the computation may result in a starting time g that does not exist, i.e. is not an element of *time*[]. In that case, the largest among all those elements of *time*[] that are smaller than the computed starting time, is assumed to be the real starting time of λ.

(6) Compute the clock bias error on the edges of the leap

$e_g^{(Window)}$ and $e_h^{(Window)}$, i.e. the clock bias error at the starting time g and at the end time h of λ (with respect to the window), are computed.

(7) Compute the leap's height

First, λ's value is computed and then the latter is used to determine λ's height: $value(\lambda) = e_h^{(Window)} - e_g^{(Window)}$ and $height(\lambda) = |value(\lambda)|$.

(8) Compare the leap's height to the detection bound

Let κ be a variable that is meant to store a real number between 0 and 1. Its value represents the confidence that $time[n]$ is not the starting time or the end time of an attack.

λ's height is compared to the detection bound of the algorithm, which is given by the parameter *detectionBound*. If the height is larger than the bound, it is likely that an attack began or ended approximately at $time[n]$ and therefore, the value *minConfidence* is assigned to κ. Else, it is unlikely that such an event occurred and hence, the value *maxConfidence* is assigned to the variable.

(9) Determine the impact of signal outages on the confidence degree

In the previous step, either the value *minConfidence* or the value *maxConfidence* was assigned to κ. The clock bias security check is, however, not meant to be a binary check, i.e. a check with only two possible outcomes. Therefore, signal outages that occurred inside the window are considered to enlarge the set of possible values for κ.

For a fixed data interval, it holds that the larger the difference between the window duration and the window length, the more (t,b)-points are missing between *time*[0] and *time*[n]. The ratio $winLength\,/\,(winDuration\,/\,dataInterval + 1)$ is a measure for the availability of data between the borders of the window and is therefore called window availability (denoted by *winAvail*). So, the lower the window availability is, the fewer clock bias values were recorded between the edges of the window.

As the clock bias naturally varies over time (i.e. also changes if no attacks take place), recording a sufficient number of clock bias values during a time interval of interest is essential for successfully distinguishing between natural variability of the clock bias and attacks that occurred during the period to be analysed. Hence, the smaller the window availability is, the larger the likelihood that a κ-value of *minConfidence* (algorithm is confident of having detected an attack) is the result of a false positive.

To counter these false positives, a correction factor that is influenced by the window availability is applied. The lower the window availability is, the more should κ be increased. Thus, κ-values that are equal to *minConfidence* are modified according to the formula $\kappa^{(new)} = 1-(1-\kappa^{(old)})\cdot winAvail.$ Thereby, $(1-\kappa^{(old)})$ is the initial confidence that *time*[n] is the starting time or the end time of an attack. This confidence degree is multiplied by the window availability, which is also a value in the range of [0,1]. The result of the latter operation is subtracted from 1 in order to get the new confidence that *time*[n] is not guilty.

(10) Output the confidence degree

The algorithm outputs its opinion on whether *time*[n] is guilty or not guilty, which basically is the value of κ. Due to the adjustments of the confidence degree in step 9, however, κ might be larger than *maxConfidence*. If so, the algorithm outputs *maxConfidence* instead of κ.

3.3.5. ALGORITHM TESTING

The detection algorithm was implemented in Java and six meaconing attacks were simulated in a row in order to test it. The receiver temperature was almost constant over the entire duration of the test and there were no signal outages. Approximately

17000 (t,b)-points arrived at the Location Assurance Provider and for all but the first *histLength* (set to 60) of these points, the algorithm was executed once in order to give its opinion on whether the concerned point is guilty or not guilty.

The values “4”, “$0.65 \cdot 10^{-7}$”, “0.05” and “0.95” were assigned to the parameters *minLeapDuration*, *detectionBound*, *minConfidence* and *maxConfidence*, respectively. The choice of these values is discussed in (Marnach, 2012). Here, we merely provide the justification of the detection bound’s value: When manually analysing leaps, it was found out that the repeater that is employed to simulate meaconing attacks causes leaps with a height of at least $0.65 \cdot 10^{-7}$ seconds (cf. Section 3.3.3). The detection bound was chosen to be equal to the latter value. This means that the bound is set according to the adversary’s assumed power and goal. The smaller the detection bound, the more false positives will occur. Therefore, it is recommended to increase the bound whenever possible.

The beginnings and the ends of all six attacks were found. Moreover, the algorithm detected two false positives, i.e. leaps that were caused neither by the beginning, nor by the end of an attack. By visualising these leaps, however, it became clear that they may also have been wrongly detected by a human observer.

Overall, the experiment showed that it is possible to detect the beginnings and the ends of meaconing attacks automatically by using the detection algorithm. However, one also has to admit that the test was conducted under laboratory conditions, as the receiver’s temperature was extraordinary stable.

4. CONCLUSION

The present section summarises the topics covered by this paper and outlines an idea that could further improve the clock bias security check.

4.1. SUMMARY OF FINDINGS

The LASP project, which is funded by ESA and realised by itrust consulting and the University of Luxembourg aims at compensating for the absence of authentication (of satellite signals towards civilian users) in satellite-based navigation systems. The Location Assurance Provider (LAP) is a component of the LASP architecture that runs so-called security checks to verify computed positions’ authenticity. These checks are algorithms that get as input data sent to the LAP by one or more user devices and that output an opinion on whether the analysed information is authentic. In this paper, we presented one of these algorithms, namely the Clock Bias Security Check. It was developed explicitly for the detection of meaconing attacks, i.e. attacks that intend to delay navigation signals.

The theoretical background that the check is based on was provided in Section 0. As meaconing always introduces delays, timing information can be considered as one of the approaches to detect this type of attack. Thereby, a receiver’s clock bias, which is the difference between the value of the receiver’s clock and a pre-defined reference time, is of particular interest.

The currently employed LASP user device was discussed in Section 2. Knowing the behaviour of the receiver that is part of this device was of particular importance for the development of the security check. Also, a possibility to simulate meaconing attacks by using a GNSS signal repeater was introduced in this section.

Details on the check itself were provided in Section 3. First, a theory on the detection of the beginnings and the ends of meaconing attacks was established. This step resulted in an attack visualisation technique that is based on the computation of regression lines. Second, the knowledge that was gained during the manual attack visualisation step was used to design and implement an algorithm that is capable of detecting the beginnings and the ends of meaconing attacks automatically. This detection algorithm represents the clock bias security check. An experiment conducted in a controlled environment confirmed that the algorithm works properly: Six meaconing attacks were simulated by using the signal repeater and the security check was capable of detecting the beginnings and the ends of all six attacks, while resulting in no more than two false positives.

4.1.1. FUTURE WORK

First test results of the clock bias security check were collected and briefly described in this document. They look promising, but it is clear that more experiments must be conducted before we can claim that the check is fully reliable. So far, the algorithm has, for example, only been used in combination with a data interval of 1 second, while it is designed to work with any data interval that remains constant over time. Besides testing the existing check and tuning its parameters, we believe that the algorithm itself could be further enhanced. One concrete suggestion for improvement is outlined in the following paragraph.

The detection bound is a parameter that has to be set by the algorithm's user, which in this case, is the Location Assurance Provider's operator. The lower the bound, the smaller are the delays that can be detected. A lower bound, however, also leads to an augmented number of false positives. How low the bound may be without resulting in too many false positives depends on the environmental conditions under which the receiver is used, especially on its ambient temperature. As the temperature often does not change abruptly, it might be possible to adjust the detection bound automatically by considering data recorded in the past. The computation of the bound could be based on the clock bias error, on the clock bias itself, or even on the clock drift, which as mentioned in Section 2.2, is mainly influenced by the receiver's temperature. In any case, it is preferable to find at any moment the lowest possible bound that leads to an acceptable number of false positives.

Also, it is planned to run the algorithm on data recorded during a real meaconing attack to make sure that the observed leaps are not merely the result of unexpected side-effects of the employed attack simulation technique.

One question that remains open is whether the clock bias security check will also work in combination with commercial off-the-shelf user devices, like smartphones. These devices' internal clocks are far less accurate than that of the currently used receiver, which poses a threat to the effectiveness of the security check.

5. ACKNOWLEDGEMENT

This work was supported by the European Space Agency under the project "Developing a prototype of Localisation Assurance Service Provider (LASP)" with contract number 4000102584-10-NL-HE. The authors would like to thank Stefan Wallner (ESA) for his suggestions for improvement concerning this paper.

6. REFERENCES

Scott L.: Location Assurance. GPS World, [July 2007], pp. 14 – 18, 2007.

Stansell T. A.: Location Assurance Commentary. GPS World, [July 2007], p. 19, 2007.

Hein G.W., Kneissl F., Avila-Rodriguez J.-A., Wallner S.: Authenticating GNSS Proofs against Spoofs Part 1. Inside GNSS, [July/August 2007], pp. 58 – 63, 2007.

Hein G.W., Kneissl F., Avila-Rodriguez J.-A., Wallner S.: Authenticating GNSS Proofs against Spoofs Part 2. Inside GNSS, [September/October 2007], pp. 71 – 78, 2007.

Harpes C., Jager B., Gent B.: Secure Localisation with Location Assurance Provider. ENC-GNSS, 2009.

Kaplan E. D., Hegarty C. J.: Understanding GPS – Principles and Applications (second edition). Artech House, December 2005.

El-Rabbany A.: Introduction to GPS – The Global Positioning System (second edition). Artech House, 2006.

Borre K., Akos D. M., Bertelsen N., Rinder P., Jensen S. H.: A Software-Defined GPS and Galileo Receiver – A Single-Frequency Approach. Birkhäuser, Bosten, 2007.

Hurn J.: GPS – A Guide to the Next Utility. Trimble Navigation, 1989.

Novick A. N.: Atomic clock accuracy needed. IEEE Potentials, Vol. 13 (2), pp. 8 – 10, April 1994.

JAVAD: DELTA Operator's Manual. August 2010, http://javad.com/downloads/javadgnss/manuals/hardware/Delta_Operators_Manual.pdf

JAVAD: GREIS Reference Guide. May 2011, http://javad.com/downloads/javadgnss/manuals/GREIS/GREIS_Reference_Guide.pdf

Vig J. R.: Quartz Crystal Resonators and Oscillators – For Frequency Control and Timing Applications – A Tutorial. November 2008, http://www.ieee-uffc.org/frequency_control/teaching/vig/vig3.ppt

Marnach D.: Implementation and Validation of Security Checks for GNSS Location Assurance – The Clock Bias Security Check. University of Luxembourg, February 2012.

Received: 2012-08-20,

Reviewed: 2013-01-16,

Accepted: 2013-01-31.