DOI: 10.15439/2016F580
ACSIS, Vol. 9. ISSN 2300-5963

Digital signing for short-message broadcasted traffic in BLE marketing channel

Jarogniew Rykowski
Poznań University of Economics and Business
Niepodległości 10
61-875 Poznań, Poland
email: rykowski@kti.ue.poznan.pl

Mateusz Nomańczuk
Billennium sp. z o.o.
Tagore'a 3
02-647 Warszawa, Poland

***Abstract*—As long as Bluetooth Low Energy (BLE) was mainly applied for broadcasting marketing information, the problem of trust of this transmission was treated as marginal. However, once the marketing channel was applied for such application as geolocation by means of BLE beacons, and e-payments, the problem of proper identification and authentication of the broadcasting device, as well as time&place of interaction, become very sharp. This problem cannot be solved by means of traditional mechanisms such as symmetric and asymmetric cryptography, due to several reasons. First, symmetric cryptography needs a redistribution of an encryption key, common for all the network nodes or at least known for the network central authentication point, and kept secret for the lifetime of the nodes. It is very problematic how to keep such multi-copied and long-lasting information secret. Second, the messages broadcasted in BLE marketing channel are restricted by length and format, making it practically impossible to use longer encryption keys widely assumed as safe. Third, BLE devices are usually very restricted according to memory amount and processing power, thus classical implementation of PKI encryption algorithms is very problematic. Fourth, there is no way to apply usual two-directional interaction to exchange some data to be encrypted, e.g., to proof directly the fact of interaction between two devices. And last but not least, time representation in small autonomous devices is quite weak, thus the hardware must be extended by some additional verification mechanisms and specialized hardware modules.**

In the paper we present a practical approach to an efficient representation of a testbed for trusted geolocation beacons broadcasting in the BLE marketing channel. The encryption is based on external co-processor and elliptic curves algorithms, which made it possible to apply shorten keys and use minimum resources of the beacon (memory, processor, energy). To prevent the attacks of "recording" type in man-in-the-middle mode (reusing the broadcasted information obtained in one place in the other place/time), the broadcasted messages include time stamps generated by attached RTC units. The idea may be applied for the other types of IoT and sensor networks to improve trust and verification of broadcasted messages.

This work was partially supported by the GOLIATH project jointly funded by the Poland NCBR and Luxembourg FNR Lead Agency agreement, under NCBR grant number POLLUX-II/1/2014 and Luxembourg National Research Fund grant number INTER/POLLUX/13/6335765.

This work was partially supported in scope of the project "Creating a platform to support management of Rego projects" co-financed by the Operational Programme Innovative Economy, action 1.4-4.1 - Support for research, development and implementation of the results, the grant agreement No. UDA-POIG.01.04.00-14-221 / 11 -00

I. INTRODUCTION

RECENTLY we observe a boom of Bluetooth devices and applications. Bluetooth (BT) communication is widely applied for short-distance networking, mainly for personal purposes, but also for some sensor networks [1]. With the introduction of version 4.1 (Bluetooth Low Energy BLE [2], recently also called Bluetooth Smart), apart traditional one-to-one transfer among two paired devices, it is also possible to generate and receive messages in the broadcast mode. Such a message, generated by one BLE node, may be read by any other BLE node, temporary present in the radio-range area, without the need of previous pairing or installation of some new software. Originally, BLE broadcast was applied for some marketing purposes. The idea was to generate periodically some short messages with advertisement of "local" products and services, to attract anybody in the closed neighborhood. This idea soon evolved towards a standard called BLE Marketing Channel. The standard concerns possible message formats, as well as physical parameters of the transmission – maximum power, message length and necessary parts (such as preamble and CRC), minimal gaps between transmissions (frequency of repetitions) etc. In a while BLE broadcast was applied to some other application areas, such as shopping and orientation in supermarkets. This idea in turn resulted in the introduction of beacons.A `beacon` is a small autonomous device, disconnected from the network, periodically (usually few times per second) broadcasting some information about itself, mainly unique identifier [3]. The identifier, in conjunction with an external database, may be used to deduce an exact geo-location of the device. As the transmission power of the device may be adjusted to current needs (at the installation time only, however), restricting the transmission distance from centimeters to a few meters, it may be assumed that all the receivers in the radio-range share the same location as long as they are able to receive proper information from the beacon [4].

The above mentioned "receivers" are any BLE devices capable of reading broadcasted messages. To this goal, any modern smartphone or tablet would apply. In case of iOS devices, there is no need for an installation of any additional software – support for BLE traffic is included as basic functionality of the operating system. In case of Android, usually a dedicated application must be installed. For Windows, according to our best knowledge, there is no BLE support as for now.

Once the BLE technology is so popular, and both hardware and software is there, it is very probably that BLE broadcast will be used in many places and for many different purposes, such as sensor networks, automatic ticketing, e-payments [5], tracking etc. However, we have to enumerate not only the advantages, but also new problems provoked by possibly mass usage of this technology. As the main target or the BLE broadcast was addressed to the advertisement, such features as trust, privacy and security [6] were not initially taken into attention. As a consequence, the standard bypasses such basic functionality as digital signing of the broadcasting BLE nodes, encryption and decryption of the message content, verification of message consistency (apart standard CRC verification) etc.

The main goal of our work is to fill the gap. In the paper we discuss some possible ways of hardware and software extensions for geolocation beacons (and similar, any broadcasting node of a sensor network) to be used for any application which requires much more level of trust. To preliminary test the idea, we propose some ways for early-prototype development, based on linking BLE devices with Windows-operated PC, to use its full potential for finding the best algorithms for the encryption and decryption of the information to be broadcasted by the beacons.

The remainder of the paper is organized as follows. First, we briefly describe BLE marketing channel and data formats for geolocation beacons, followed by a discussion on the problem of a lack of efficient cryptography method for this sort of devices. Second, we try to formulate basic requirements for such method, to be applied mainly for digital signing of the broadcasted information. Then, we propose an environment for testing several solutions, with a prototype testbed based on AVR processors [7], BLE modules and some PC-based simulators of encryption modules. Finally, we provide some conclusions and show the directions of future work.

II. BLE MARKETING CHANNEL AND GEOLOCATION BEACONS

As mentioned already, a beacon uses so called BLE marketing channel to disseminate some information about itself, mainly unique identifier. The channel is characterized by some restrictions, especially introduced to minimize energy consumption and extend battery life. In particular, in any case message size cannot exceed 47 bytes, and certain time gap between messages must be preserved, making it possible to transmit only a few messages per second.

The above requirements substantially reduce the possibility to directly apply encryption for the broadcasted messages, for at least two reasons: limited data length, and limited possibility to mix both encrypted and non-encrypted data.

To better understand these restrictions, we must describe the data format that is used to disseminate messages in BLE marketing channel. Each message, of length 47 bytes, is composed of (Fig 1):

- a preamble, always equal to 0xAA,
- channel address (in turn always equal to 0x8E89BED6),
- data packet, composed of a header, additional address field, and unique device identifier (in case of the most popular iBeacon standard[8], for other standards such as AltBeacon [9], Radius Networks, Google beacons [10] etc., similar restrictions apply),
- strength of the radio signal (power level),
- standard Cyclic Redundancy Check (CRC) value to automatically detect and correct transmission errors.

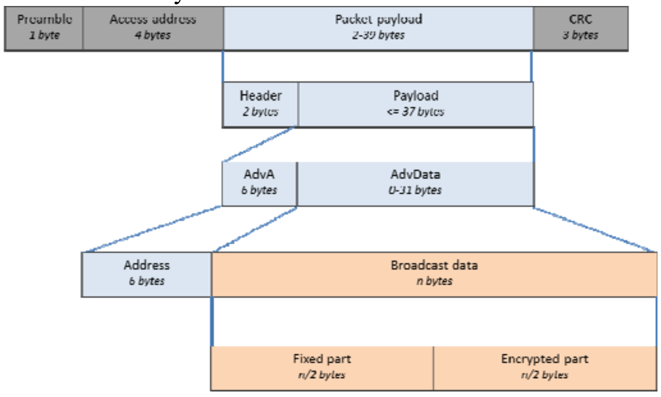

Fig.1. Single frame for BLE marketing channel, iBeacon standard

As it may be seen, only 31 of 47 bytes composing the message may vary, while the rest is reserved for the purposes of BLE marketing channel standard. This means that the maximum length of the encrypted part is limited to 248 bits. It is widely assumed that the encryption key must be shorter than the message to encrypt. Thus, in this case the encryption algorithm should use keys of minimum length, which are not treated as safe now. Note that parting the encrypted data into several messages is also very problematic, due to the fact the broadcasting takes place only few times per second, thus the reception of the whole message would be incredibly long. Note also that if we would like to transfer in the same message both encrypted and non-encrypted part (for the purposes of digital signing), the encrypted data length is limited to 20 bytes, thus 160 bits at the most.

Taking into account the above restriction, we decided to applied the newly proposed encryption algorithm based on elliptic curves (ECC) [11]. This algorithm is effective for the substantially shorten keys (in comparison with classic algorithms such as RSA [12]), its 160-bit long key provides similar encryption power as 1024-bit RSA key. The problem is the ECA algorithm is hardly applicable to the limited hardware/software of the beacons. Even if successfully implemented with limited memory and CPU resources, it would consume substantial amount of energy for the computations. Thus, for the encryption a specialized chip should be applied, to (1) encrypt the data in parallel with the execution of a beacon program, and (2) to minimize energy consumption with the encryption process implemented at hardware level.

So far, only limited number of such specialized chips is available on the market; however, this restriction is expected to be relaxed. In parallel, several basic libraries have been proposed recently to apply ECC algorithms for the encryption, as DSA/RSA replacement. These libraries are available for popular programming languages such as Java and C/C++/C#. In the next section, we broaden the discussion on this topic, addressing also some implementation and organizational issues related to the usage of trusted (signed) information broadcasted by the beacons.

III. Theoretical Aspects of Encryption for Broadcasted Messages

In the classical application of a beacon to geo-location marking, each beacon broadcasts only its unique identifier, and no encryption is applied. Also, no time is to be represented and broadcasted – the broadcasted signal is the same for the lifetime of a beacon (usually from a few months to a few years). As such, the signal is not resistant to any man-in-the-middle attacks, such as capturing a message from a beacon at certain place and re-broadcast this signal using a fake transmitter at another place. Moreover, such attack seems to be trivial using popular devices such as smartphones with BLE units. It may be expected that sooner or later attacks of this type will take place, especially at the shopping centers and popular tourists points, for which the beacons are widely used as basic location markers.

It is clear that the beacons should be extended by some verification mechanism. And it is quite evident that the cryptography should be applied to this goal. The question is – which kind of cryptography and which algorithms?

As for the cryptography type, we deal with symmetric and asymmetric cryptography. The base for the first is to distribute some encryption keys, usually common for all the nodes in a local "network" (it's hard to say the broadcasting-only nodes create a full network, however, we will use the term "network" to point out the fact the nodes are communicating using networking mechanisms). The problem with such distribution and storage is to keep the keys secret. As the beacons are fixed and cannot be changed/updated, and they have no bi-directional network connection, it is very naïve to think the key would not be unfolded, and thus some fake devices would be added to the "network". Moreover, as most of nowadays smartphones are equipped with a BLE unit, it would be possible to run an application pretending to act as a beacon with minimum effort. For this reason, we should abandon the idea of using symmetric cryptography for beacon network.

The asymmetric cryptography seems to be much better candidate [13]. As the private key is possibly generated at the installation time inside the device and never accessible outside, there is no way to unfold it. And even the device is cracked by the hardware, it is possible only to falsify this device only, while the other devices are still safe and trusted [14]. The public key may be either propagated at the installation time to the operator, to be stored in an external database indexed by the device unique identifier [15], linked to IP address [16], or even periodically broadcasted by the beacon together with its identification data.

However, several problems arise while trying to implement asymmetric cryptography for BLE marketing channel, these are discussed below, with some proposals towards efficient implementation.

First, as already mentioned in the previous section, BLE marketing channel is very restricted according to total length and format of the broadcasted messages. Second, majority of information from each message is fixed due to the requirement of the BLE marketing channel standard. As a result, only small fraction of the message is variable, and only a small part may contain encrypted data. Thus, it is not possible to apply classical encryption algorithms such as RSA, for which the basic requirement is to encrypt the information longer than the encryption key. A reasonable key is 1024 bits long, while the minimum encrypted message length is something like 128 bytes (not counting fixed elements of each message such as headers and checksums), which is at least three times as much as the maximum length of a message in BLE marketing channel. One may say that a single message may be cloned to form a longer message, and such a long message may be divided into a few smaller messages, transmitted into pieces and finally relinked and extracted at the receiver side. However, due to energy savings, minimum period between succeeding transmissions is counted in hundreds of milliseconds. Thus, transmitting the whole long message would take a second or even more, assuming there will be no transmission errors. As the useful radio-distance for a beacon is sometimes counted in centimeters, one would have to stop and wait near each beacon to get the whole encrypted message. This is unrealistic, moreover, usually we do not know exact locations of the beacons, thus having no information where to stop and for how long.

So, if RSA (and similar solutions) cannot be directly applied, we have to take a look for its replacement. Elliptic Curve algorithms (ECC) seems to be a good candidate to this goal [11]. With their 160-bits long keys they offer encryption power comparable with 1024-bits RSA [12]. Once a minimum length of an encrypted message is counted down to 20 bytes only, ECC encryption fits the maximum length of BLE message.

In addition, ECC algorithm is much less demanding according to the memory and CPU requirements in comparison with RSA [12]. Note that when we have to e.g., double the memory amount, we usually shorten the battery life by the same factor. Thus, for the autonomous beacons, for which the lifetime should be counted in years rather than months, it is extremely important to avoid complex computations and mass usage of operational memory. However, the problem is that typical processors applied for beacons and similar IoT/sensor network devices are very restricted. E.g., popular AVR family offers 8-bit processors with 2 kB of operational memory as the base for small battery-operated devices. Even if some AVR processors are 32-bits machines with hundreds of kilobytes of memory, it is still not reasonable to use such hardware for ECC computations. As stated by many researches, it takes typically several hundreds of milliseconds to perform a single encryption for a short message while using AVR-related hardware [7]. Our experiments also showed that efficient implementation of any asymmetric encryption algorithm for AVR processor, even if possible, takes too much time and leaves almost no place in the memory for the other code, needed at least to control the operation of BLE unit and message composition.

The solution is to apply a separated hardware module dedicated to encryption tasks. We tested such modules, but found their usage very difficult while applying for AVR-based nodes. Unfortunately, these modules are hardly re-programmable, and there is limited number of libraries for popular AVR software-design platforms. It looks like a lot of work should be done in order to propose a more practical solution. Anyway, we clearly see this is the right way, and we would like to test this way even if the external, fully programmable encryption modules are not ready yet. Our approach towards such a testbed in unfolded in the next section.

Finally, we have to discuss the strategy for direct beacon verification, and indirect verification of time/place of the interaction with the beacon. Usually, to prove the fact of such interaction, bi-directional transmission is used based on exchange of some encrypted and to-be-encrypted information [16, 17, 18]. For example, to prove the fact of being in the radio-range with a device, another device near-by composes a random message, encrypts it with public key of the device and sends to this device. The device, using its private key, decrypts the message, in turn encrypts it with the public key of another device and sends it back. Such double encrypted/decrypted message may be a proof of the cooperation of these two devices. If in addition encrypted time-stamps are exchanged rather than randomized messages, one is able to prove not only the fact of transmission, but also its location in time, proven by both parties.

As the above schema is based on bi-directional traffic, it cannot be applied for the broadcasted-only system. Instead, a different approach must be applied, aiming in using different broadcasted messages for subsequent transmissions. The messages must differ in such a way in reasonable time there will be no possibility to use the same device twice. As the geo-location message in the BLE marketing channel is sent few times per second, and the lifetime of the beacon is counted in years, there should be at least approximately 30 million of different combinations, which stands for a "long integer" value – not a problem even for a very small microcomputer. Moreover, this number may represent time (e.g., as number of seconds passed since the moment of installation), acting as a time stamp for proving the fact of interaction and never re-used. For better quality, Real Time Controller (RTC) module may be added, counting an reporting seconds (or even milliseconds, if needed) with an error not exceeding few seconds per year, which is acceptable for most of the applications of geo-location beacons.

In the next three sections we depict a testbed implementing a network of trusted geolocation beacons, composed of three levels:

- beacon-simulator level, based on Arduino hardware and software with dedicated BLE transmission module,
- central monitoring and encryption node, based either on extended Arduino controller with hardware encryption module, or a PC with encryption software,
- application level, aimed in using Android-controlled devices to test the behavior of the trusted beacons.

IV. Beacon simulators

In general, we were not able to directly apply any of market solution, i.e., commercially available beacons, to be used in the trusted mode. The reason is the beacons are closed solutions, with fixed behavior and with no possibility to re-program to suit our needs for the encryption and trust. Thus, since the very beginning we decided to use beacon simulators instead. A simulator is based on Arduino microcontroller with BLE transmission unit, real-time clock and encryption module attached (Fig. 2).

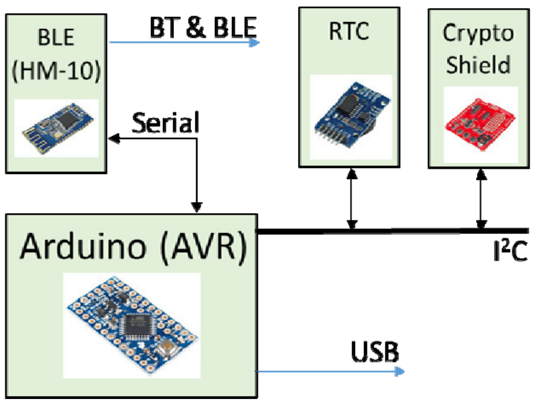

Fig.2. Beacon simulator

We have to point out here that AVR-based hardware, Arduino included, is very restricted according to efficient ways of code preparation and testing. Usually, the code is prepared using a PC-based compiler and then send to the target module by a communication link (typically Bluetooth or USB). There, the code is to be executed, usually with very limited possibilities of tracking and debugging. As a result, proper implementation of sophisticated tasks, such as asymmetric encryption protocols, takes a lot of time and programmers' efforts and requires high programming skills.

The operation mode of a beacon simulator was the following. First, RTC time was fetched by AVR to get unique time stamp. The stamp changed every minute, which was sufficient enough for most applications. Together with unique identifier, the time stamp formed a message to be digitally signed and broadcasted. The signing was realized by means of asymmetric cryptography algorithms and a private key stored in the memory. The encrypted data were then available to read and decrypt using a public key, in turn accessible from an external database, as correlated with the beacon identifier [19].

As it may be seen, the broadcasted and encrypted data make it impossible to perform the "record and play" attack, when someone simply collects the broadcasted info at certain place and time and further present it as authorization data for another place/time. As mentioned before, for untrusted beacons such an attack is trivial with nowadays hardware and software, e.g., using a typical smartphone. As a result, in the case of our proposal a level of trust for broadcasted information is substantially increased. However, the received information still may be exchanged with some other devices, to simulate the presence at given place/time. To restrict such way of cheating, the application receiving the broadcasted messages in turn should be digitally signed and installed in the trusted way. By linking the trusted broadcast and the trusted application receiving this broadcast, one may prove the fact that the receiving device was really present at given place/time, able to receive and memorize the broadcasted information there.

The most important problem related with the implementation of a trusted beacon is related to the efficient implementation of the encryption. As for our tests we assumed an application of AVR processors and cheap modules such as Arduino microcontrollers, in general we had three possible ways to implement the encryption algorithms:

1/ directly in AVR code (C/C++ and assembly language),
2/ by means of specialized hardware modules, directly connected to one of AVR communication links,
3/ as a Java-based library to be executed at PC side and contacted via AVR communication links.

The first way, according to the very limited hardware/software resources of small microcontrollers, seems to be impractical. Even if we successfully implemented and tested several encryption algorithms (not only asymmetric-cryptography algorithms, also e.g., TEA symmetric-cryptography algorithm for assuring secrecy and privacy [20]), we found that the amount of RAM and ROM memory is not sufficient to include, apart the encryption algorithm, also some additional code to deal with some other tasks (such as timings, broadcasting, BLE transmission, etc.). Remember that AVR-based solutions are not based on underlying operating system, thus all the tasks to be performed by the node must be included directly in the program code, starting from such simple procedures as blinking a LED, and finishing on bit-by-bit serial communication and encryption algorithms.

Then, we began to look for some specialized modules to perform the encryption at hardware level. We discovered several proposals, among these the CryptoShield with ATSHA204, HMAC256 and ATECC108 chips seemed to be very promising [21]. Together with specialized chips for traditional RSA encryption and Real-Time Clock RTC add-on, and the popular I^2C communication link, the module at the very first view was ideal for our goals. However, we soon detected several drawbacks, making it questionable to apply the module for beacon network. First, the module consumes a lot of energy, even if not used frequently, due to the fact the module is equipped with its own microcontroller (Atmel Trusted Platform Module for CryptoShield version, and ATmega328 for CryptoCape). Second, the size of the module is ideal for basic Arduino controllers such as Uno and Mega, but seems to be much too big in case of the smallest controllers to be used for beacon implementation, not to say about the need of additional connectors to map the pin-out. Third, even if advertised as compatible with the smallest 8-bit controllers, the module in the matter of fact is efficiently working only with the most powerful Arduino boards (32-bits CPUs and Linux-based control), in turn increasing energy consumption and limiting battery life.

V. Architecture of a Testbed

To bypass the implementation problems mentioned in the previous section, we decided to slightly change the architecture of the whole beacon network. We went to the idea of a single cryptography unit to work for several beacons, with autonomous power supply and trusted communication links with the beacons (Fig. 3). For the initial test, we applied standard wired connections (serial transmission in UART mode, at 115200 Bd). The wired connections were also used to provide power supply for the beacons. We see that the wired solution is certainly not the target one, but as for testing and validating the whole system this solution is much better than a set of wireless connections, fixing (1) the problem of code updates – via serial/USB links, (2) the problem of power supply, and (3) the problem of radio-transmission conflicts and errors.

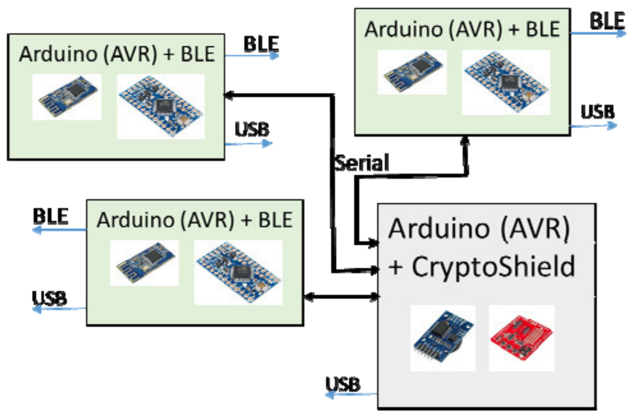

Fig.3. Architecture of the beacon network with single point of encryption implemented as specialized hardware module

The cryptography unit served not only for the pure cryptographic purposes, it also acted as a clock synchronizer (providing RTC data) and activity monitor – as all the beacons periodically quoted for encrypted time stamps, it was quite natural to collect these requests and to report them elsewhere.

Once we wanted to test the timings for the preparation of the encrypted data, and to compare several cryptographic algorithms, we found that the cryptographic module is hardly applied to this goal, mainly due to the limited re-programming possibilities. Thus, we again change the architecture of the testbed, replacing the cryptographic module with a PC serving as a central cryptographic server (Fig. 4). The serial connections to beacons were replaced by standard USB links, and the PC also served as the power supply for the beacons. The server was implemented in Java, with a help of Java Serial Connectivity (JSSC) library and certain cryptographic libraries, including ECC algorithm. Then, we were able to test several implementations of ECC-based encryption, not to say about detailed measurement of timings and the estimation for global energy consumption.

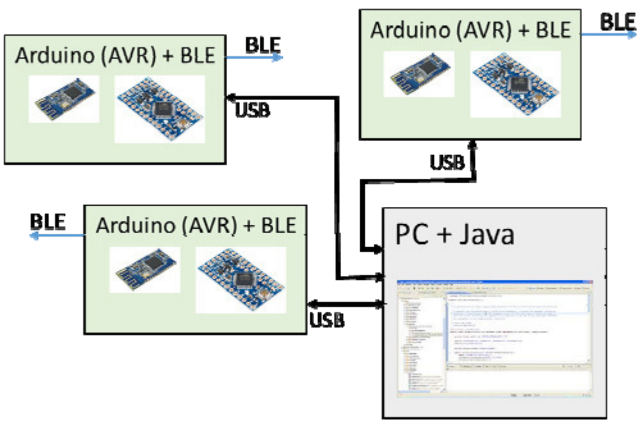

Fig.4. Architecture of the beacon network with single point of encryption implemented as PC-based server

VI. APPLICATION LEVEL

Finally, we intended to test the possibility to directly access and monitor the beacons via BLE transmission to be controlled from a portable PC, the solution which is much more practical and efficient than testing on limited Android or Apple devices. Unfortunately, there is no BLE support for a PC running Windows, even if the hardware is there. Thus, to this goal, we used AT mode of the newest BLE-to-USB micro converters, namely HM-10 module [22]. With this module, it is possible to enable so called "directory" mode to detect all BLE modules in the neighborhood. The HM-10 module was directly connected to a USB port of a PC, and the AT commands were applied for BLE monitoring and the detection of signal-strength info. The collected data were processed by a Java program. As a result, it was possible not only to validate the network of beacons, but also to undertake some tests for beacon visibility, possible distortions (due to weather, crowd, other electronic devices in the closed neighborhood, etc.), and many more. Note that, even if BLE transmission is implemented for the newest Android-based devices and Apple phones/tables, the monitoring software is usually very restricted and mainly aims in displaying the most probable location of a beacon at the screen. Our monitoring system provides much more data and may be used for the comparison of different situations, cases, places, etc., not to say about the verification of trust based on encrypted broadcast from the beacons.

HM-10 module was also found to be well adjusted to the communication with Arduino and other AVR-based controllers – this module was finally applied for the tests as a basic BLE unit for each trusted beacon. Here we have to state once again that any of the existing hardware solutions for beacons could not be applied, as there is not a single possibility to change the broadcasted data cyclically (e.g., every second) for a traditional beacon. Again, we applied a combination of AT commands and BLE broadcasting, controlled by underlying Arduino board, to convert the HC-10 module into an efficient beacon simulator.

II. Final conclusions and future work

The testbed architecture described above is the very first step towards an implementation of the target trusted beacon network. As for the future work, we plan:

1/ to measure the efficiency of several encryption algorithms,

2/ to estimate energy consumption due to the encryption tasks,

3/ to test the possibility to provide better cryptographic algorithms and chips to be included in the beacons, especially to find the optimum parameters for ECC encryption algorithm,

4/ to eliminate the need for cable connections, replacing them e.g., by periodic BlueTooth transmission in a safe master/slave mode with traditional pairing,

5/ to validate some other BLE communication modules, as soon as they are available on the market (so far, only HC-10 module seems to be useful for both beacon simulation and detection).

We must also state the fact that the approach described in this paper may be applied to any sensor/actuator network, not only for geolocation beacons, to increase the level of trust for broadcasted messages. Although BLE marketing channel and broadcast are rarely used towards this goal, this is a generic approach that is potentially of great interest for the designers of sensor networks, networks for "intelligent" places and buildings, including "smart cities"[23], applications of Internet of Things [24] and Services [25], and many more.

The results of our work towards trusted geolocalization beacons resulted in a PL/EU/US patent application entitled "Trusted geolocation beacon and a method for operating a trusted geolocation beacon", currently at early-registration phase.

References

[1] What is Bluetooth technology?, http://www.bluetooth.com/Pages/what-is-bluetooth-technology.aspx, 2016

[2] Bluetooth Low Energy Technology, https://www.bluetooth.com/what-is-bluetooth-technology/bluetooth-technology-basics/low-energy, 2015

[3] Beacon Tech Overview, Estimote documentation, http://developer.estimote.com online: 21.10.2015

[4] Bluetooth® Low Energy Beacons, Texas Instruments materials, http://www.ti.com/lit/an/swra475/swra475.pdf, 2015

[5] P. Boddupalli, F. Al-Bin-Ali, N. Davies, A. Friday, O. Storz and M. Wu, Payment support in ubiquitous computing environments, in: Mobile Computing Systems and Applications, proceedings of Fifth IEEE Workshop on. IEEE, 2003

[6] Roberts, P. F. Internet of Things Demands New Social Contract To Protect Privacy. https://securityledger.com/2013/09/internet-of-things-will-force-choice-between-privacy-control/, 2013

[7] Atmel AVR 8-bit and 32-bit Microcontrollers, Atmel documentation, http://www.atmel.com/products/microcontrollers/avr/, 2016

[8] iOS: iBeacon technology overview, Apple documentation, https://support.apple.com/pl-pl/HT202880, 2015

[9] AltBeacon − The Open and Interoperable Proximity Beacon Specification, http://altbeacon.org, 2015

[10] Mark up the world using beacons, Google documentation, https://developers.google.com/beacons/, 2016

[11] Elliptical Curve Cryptography (ECC) Definition, TechTarget reports, http://searchsecurity.techtarget.com/definition/elliptical-curve-cryptography, 2015

[12] RSA cryptosystem, from Wikipedia, https://en.wikipedia.org/wiki/RSA_(cryptosystem), 2016

[13] G. S. Quirino, A. R. L. Ribeiro and E. D. Moreno, Asymmetric Encryption in Wireless Sensor Networks, http://www.intechopen.com/books/wireless-sensor-networks-technology-and-protocols/asymmetric-encryption-in-wireless-sensor-networks - comparison of PKI algorithms and timings, 2016

[14] Adams, C., & Lloyd, S. Understanding PKI: concepts, standards, and deployment considerations. Addison-Wesley Professional, ISBN 978-0-672-32391-1, 11–15, 2003

[15] L. B. Oliveira, D. Aranha, E. Morais, F. Daguano, J. Lopez, and R. Dahab, Identity-Based Encryption for Sensor Networks https://eprint.iacr.org/2007/020.pdf, 2016

[16] Willey W.D., Device Authentication in a PKI, US Patent 8,661,256, 2014

[17] Troxler R.E., Methods, Systems and Computer Program Products for Locating and Tracking Objects, US Patent Application US 2015/0088452, 2015

[18] Balfanz D., Lopes C., Smetters D., Stewart P., Wong H.C., Systems and Methods for Authenticating Communication in a Network Medium, US Patent US 8,156,337, 2012

[19] Rykowski, J., and M. Nomańczuk, Geolocalization beacons – a new way of position determination inside buildings, in: Drives and Control, vol. 12 (200) , Druk-Art. Press, 2015 (in Polish).

[20] Tiny Encryption Algorithm (TEA), from Wikipedia, https://en.wikipedia.org/wiki/Tiny_Encryption_Algorithm, 2016

[21] SparkFun CryptoShield, https://www.sparkfun.com/products/13183, 2016

[22] HM-10 Bluetooth module datasheet, https://www.seeedstudio.com/wiki/images/c/cd/Bluetooth4_en.pdf, 2016

[23] H. Chourabi, T. Nam, S. Walker, J. R. Gil-Garcia, S. Mellouli, K. Nahon, T. A. Pardo, H. J. Scholl (2012), Understanding Smart Cities: An Integrative Framework, proc. of 45th Hawaii International Conference on System Sciences, DOI 10.1109/HICSS.2012.615, 2014

[24] Atzori, L., Iera, A., & Morabito, G. The Internet of Things: A survey. Computer Networks 54 (15), 2787-2805, 2010

[25] Towards the Internet of Services, CORDIS Software & Service Architectures and Infrastructures, http://cordis.europa.eu/fp7/ict/ssai/home_en.html, 2015