Random Beacon for Privacy and Group Security

Aleksi Saarela*
Jan-Erik Ekberg†
Kaisa Nyberg‡
*Department of Mathematics, University of Turku, Finland, Email: amsaar@utu.fi
†Nokia Research Center, Helsinki, Finland, Email: jan-erik.ekberg@nokia.com
‡Department of Information and Computer Science, Helsinki University of Technology, Espoo, Finland
and Nokia Research Center, Helsinki, Finland, Email:kaisa.nyberg@tkk.fi

***Abstract*—Most contemporary security mechanisms and protocols include exchange of random or time-variant nonces as an essential means of protection against replay and other threats or as a seed for randomness. In many cases, it would be beneficial to have such nonces available from a trusted common source, such as a satellite. The goal of this paper is to present a protocol by which a loosely connected network of devices can agree on a common piece of randomness, and show how it can be applied to improve efficiency of a privacy protection system and group session key exchange for PAN/LAN devices.**

I. Introduction

Many security mechanisms and protocols can benefit from a time-dependent beacon (nonce) from a trusted common source, e.g., a satellite. In particular, it can be used as a freshness guarantee for communication protocols, but also for other applications that need randomization. However, the existence of such omnipresent beacons cannot be set as a requirement in typical networking standards.

In this paper our particular concern is the privacy solution of the Ultra Low Power Bluetooth (aka Wibree) radio system for PAN/LAN networking [1], where public anonymous addresses are computed from the true identity and a randomizer using a cryptographic function. Given a public address a device must search through all identities known to it to find if the public address belongs to a device known to it. If all devices would use the same randomizer to compute their public addresses, parsing the list of known devices once would be sufficient. Hence the privacy solution for the Wibree radio would greatly benefit from the existence of a common trusted beacon.

In the lack of a common beacon an alternative approach to solve the problem would be that the devices agree on a common nonce among themselves. The main contribution of this paper is a protocol by which a loosely connected network of devices can agree on a common piece of information, which satisfies the security requirements for the particular application of Wibree address privacy. We investigate in the light of small simulations three different variations of the protocol. Our preliminary results presented in this paper are promising. We conclude that such protocols can be useful not only for improving the efficiency of the Wibree privacy solution but also for deploying other security and privacy applications in PAN/LAN and ad-hoc networking that rely on beacons.

The rest of the paper is organized as follows. In Section II, we examine earlier solutions in the domain of location privacy, i.e., solutions to the specific problem that static PAN/LAN link addresses of mobile devices like laptops and mobile phones can be used for tracking their location, and present the Wibree privacy system in Section III. The beacon solution is presented in Section IV and the algorithm is given in Section V. The security is analyzed in Section VI. Finally, simulations are presented in Section VII.

II. Random Beacon, Motivation and Earlier Work

To the best of our knowledge, publicly available shared randomness has previously been suggested for use only in the following context. In 1990, Ueli Maurer proposed an information-theoretically secure key-exchange protocol, which allows two parties receiving the output of a random source, e.g., a satellite broadcasting random bits, to agree on a secret key by public discussion [2].

In the lack of a trusted common random source, users of cryptographic protocols provide by themselves the necessary random nonces which are then one by one in a well defined order taken as input to the steps of the protocol. An alternative approach to solve the problem for certain practical applications would be that the devices create a common random beacon among themselves. In short, the idea is that active devices in a local network transmit seeds that are taken as input to a joint random beacon generation algorithm. Then the following security requirements can be identified:

1) The device's own contribution is used as input to the computation of the beacon.
2) No device can force the beacon to take on a given value, or more generally, to have some given relation with previous beacon values.

In this paper, two potential applications of a random beacon will be presented. In both cases, efficiency of the application can be significantly improved compared to existing solutions. We will also see that the two security requirements stated above are sufficient to encounter the threats against which individual random nonces are used traditionally.

Having all devices broadcasting seeds may cause an intolerable traffic increase in big networks. The beacon formation would also require a single node to receive all other seeds, but it cannot be expected to catch all broadcasts of all other nodes at once and without transmission errors. In short, an

ideal beacon formation algorithm should from a computation and networking perspective also

1) Provide the same beacon value (or a beacon value from a small set of simultaneous beacon values) in every node as quickly as possible.
2) Minimize the transmission need.
3) Be robust in the presence of heterogenous, close-to-disjunct network topologies, and in the presence of network volatility.
4) Minimize the need for node-local state (memory).

The problem of efficiently distributing the seed information among nodes is closely related to the issue of distributing routing information efficiently in ad-hoc networks. Although the de-facto solutions for routing in ad-hoc networks rely on reactive routing (not suitable for the problem at hand), there is a wealth of research towards pro-active routing for ad-hoc and sensor networks, examples include e.g. [3] and [4]. Data aggregation using Bloom filters is a known property, and has been proposed e.g. for efficient sensor authentication in information flooding applications [5].

Location privacy has previously been provided by randomized link addresses that are established in an explicit authentication protocol. In a WLAN context, an authenticated session is typically set-up based on randomised MAC addresses ([6], [7]). In Bluetooth, a design where advertised addresses are short-lived, and identities are revealed only during pairing has been proposed [8][9]. Also GSM and 3G networks provide temporary identifiers (TMSIs) to protect users against omnipresent tracking. This approach works for networks where the function of the access point has no privacy constraint. It is, however, of limited use in PANs, where the finding of the intended peer device or service using a private address would then imply establishing a connection to all reachable devices to resolve their respective identities.

To provide location privacy for all PAN devices the Wibree radio system has developed a different approach described in [10]. Each device holds an identity root key which is given to peer devices at pairing. A random address of the device then consists of two parts, a random number r and a tag $E_k(r)$ of the random number r computed using the device's identity root key k.

Suppose a device D knows private keys $k_1, k_2, \ldots$. If it sees an address (s,t) and wants to find out if this belongs to one of the devices $A_1, A_2, \ldots$, it must calculate $E_{k_1}(s), E_{k_2}(s), \ldots$ and check if one of these matches t. If it sees another address (s',t'), it must do all this over again.

Suppose next that all the devices in the area would use the same value of s. Now the device A can calculate $E_{k_1}(s), E_{k_2}(s), \ldots$ in advance, just once and store them. When it sees an address (s,t), it can check if there is a match with any of the precalculated values. This will speed up the process considerably.

A second application that might benefit of using random beacons is group session key generation. The formation of a group is typically done in the application layer and a group master key is established at the group set-up phase. In IEEE 802.11 WLAN networks as in similar MAC-layer standards group session keys are still specific to transmitters and are established by running the pair-wise four-way handshake. It means that each transmitter device has to run the four-way handshake with all the recipients. With a joint random beacon the group session key can be established without any handshake procedures at all – each device just computes $f_k(s)$, where f_k is a key derivation function with group master key k and s is the beacon.

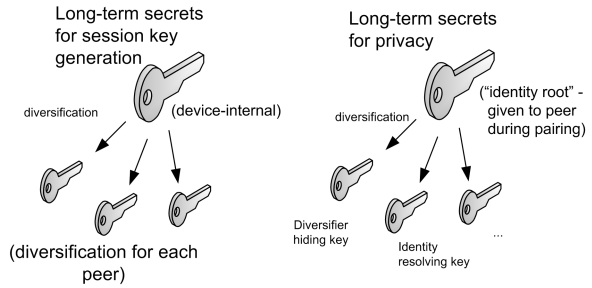

Fig. 1. The two key hierarchies of Wibree

III. WIBREE - A REFERENCE

We shortly present the Wibree radio, an example of a technology where the concept of private addresses is being introduced. Wibree devices use different addresses for connection establishment and ongoing sessions. Devices advertise their presence and are respectively discovered based on 48-bit addresses that may be either static, device-specific global addresses or private addresses. When a connection is established, the communicating devices switch to use shorter, 32-bit temporary addresses, which do not change during a connection. The privacy concept only targets the 48-bit discovery addresses.

To minimise host code size and support slow host processors, privacy functions as well as key management operations use an AES hardware block whenever possible - for encryption, key diversification and for constructing cryptographic authenticators.

The keying in Wibree is based on two disjoint key hierarchies, see Fig. 1, one for confidentiality and one for privacy protection. In session key establishment for a protected connection as well as for privacy features, only the long-term key sets of the target device will be used.

The privacy system is outlined in [10] and supports privacy functions for both the connecting party (initiator) and advertisers. However, only advertising devices (targets) need to manage their private addresses in a timely fashion. Typically, a device changes its private address at boot, after every connection, and periodically, say every 15 minutes if it is continuously advertising

The 128-bit *identity root* k_A forming the basis of the address construction is released as part of a pairing process to peers that need to connect to the device. A device may choose to advertise with one of several possible identities - several at a

time if the link layer hardware supports connection attempts to one of several identities. A private address is a one-way function construct, $addr_{priv} = (rand_{0...23}, E_{k_A}(rand_{0...23}))$.

The concept described above is easy to implement, but has some obvious drawbacks. In cases where a device looks for a specific other device, the identification amounts to a linear search against all seen addresses, with one AES operation done for every address resolving operation. This is not too different from the case where a fixed address is recognised from a list of addresses returned as a result from a scan - the AES hardware is estimated to calculate a block in less than a millisecond so no significant delay should occur. However, for situations where any recognisable device from "the address book" of size m is looked for from the set of reachable devices n, the matching problem increases in difficulty to $m \times n$ - for larger sets inefficient both in terms of search times and energy consumption.

IV. Beacon framework

In this section, we define a framework for beacon generation which can be used to describe the several alternative mechanisms that are discussed in further sections. We assume the participating nodes to be part of the same (local) networking environment, and all participants that contribute to the common beacon will be able to assert its freshness, and thus its applicability for e.g. private addressing or group session key exchange.

We define a beacon contribution of a device A to be of the form (r, L, B), where r is a random seed selected by A, $L = (r_1, \ldots, r_l)$ is a list of some seeds of other devices, B is a set containing all random seeds A has seen. Initially, the list L is empty and the set B contains just r.

Suppose that at some given instance A has received contributions (r_i, L_i, B_i), $i = 1, \ldots, m$. It will select a maximum l of the random seeds $r_1, \ldots, r_m$ (or all of them, if $m < l$) and construct the list L of those. Note that the values in other lists L_i are not included. In the set B it will add all the values $r_1, \ldots, r_m$ and all those in the lists L_i. The values that were in B before will also remain there. This way the set B will be an aggregate of all the random seeds A has seen either directly or in the lists L_i, including its own r, but not including those only in the sets B_i. The list L can change totally each turn, the set B only grows.

The purpose of the L-lists is to speed up the diffusion of the random seeds and remedy problems related to network topology and incomplete message diffusion. It is possible to set $l = 0$, i.e. make the lists empty. The simplest way to select the elements of L is to choose them randomly. A more intelligent approach is to prefer those random contributions that seem to be missing from many B-sets of other devices.

A device need not contribute if enough devices are already contributing and the resulting value is acceptable to the device (it has not been used before). If a device is broadcasting alone, it may add in its L-lists additional random values to artificially enlarge the entropy of the resulting seed. The same approach can be made for two or three devices, but devices should soon stop this activity as the network grows.

A. Implementing Set B

We propose to implement the set B as a Bloom filter. It has a clear advantage for this application, since it is independent of the order and the number of times each seed is added to it. A basic Bloom filter construction employs k different hash functions, each of which maps an element to an ℓ-bit hash code, and a memory array of size $m = 2^\ell$. To add an element, compute the values of each of the k hash functions to get k array positions. Set the bits at all these positions to 1. If the number of elements to be added is n, then the value of k that minimises the probability of false positives is $k = \frac{m}{n} \ln 2$, which gives the probability $0.6185^{\frac{m}{n}}$ of false positives. For example, if $\ell = 10$ and $n = 100$ then $k = 7$ and the false positive probability is about 0.008.

The selection of the hash-function and parameter lengths for a particular application depend on the security requirements and the communication model.

B. Beacon construction alternatives

We present three methods to generate the random beacon s as an aggregate of B-sets available to the node. Suppose a device has heard the sets $B_1, \ldots, B_k$ on some turn and its own set is B and random contribution r. For Bloom filter aggregation, we present three methods to generate a source value S that collects all the entropy to be used for the beacon s, which can be computed as a hash function of S presented as a bit string.

1) $S = B$. This is the simplest strategy.
2) $S = (B \cap B_1 \cap \ldots \cap B_k) \cup \{r\}$. The device must include its own r to make sure it has an effect and to make sure that the set S is not empty. This strategy attempts to use only those random contributions that are known to sufficiently many devices.
3) $S = B \cup B_1 \cup \ldots \cup B_k$. This strategy attempts to use as many random contributions as possible.

Suppose no devices enter or leave the network and every device has some possibility to see every other device. At some point, which does not depend on the above strategies, all the devices have heard all random contributions and their sets S will be all the same. However, the union strategy can possibly reach common S before this. The strategy $S = B$ with $l = 0$ is the most primitive way to form the seed: a device simply uses the random numbers of those devices it has seen.

The impact of the different beacon construction alternatives is easily visible in the simulations of section VII.

C. Contribution aging

If the network is highly volatile (devices enter and leave) the sets B quickly increase in size. Some mechanism to forget old seeds must exist.

If the devices have enough memory, they can store every random contribution they have seen along with the time it was last seen. If some r has not been seen within certain time, it

will be removed. A more space-efficient way is for the node to clear its B every now and then (and re-enter its own r). This clearing can either be time-depedent or be triggered by the sizes of the S or B - sets. The parameter selection is an optimization issue, since on one hand the clearing operation will cause some disturbance in the beacon generation, but infrequent clearing may lead to devices having diverging B-sets for extended periods of time.

To illustrate the problem, suppose that there are 11 devices currently in the network, arriving at time units $0, 40, 80, \ldots, 400$. Also assume that 10 devices arrived at time moment 0 and left at time moments $20, 60, 100, \ldots, 380$, and no B-set is cleared. If every device was able to gather all the random seeds in the network in no less than 20 time units, then all the B-sets are consistently different, and if the strategy $S = B$ is used, so are the seeds. With union strategy the seed of a device is determined by the oldest device it sees on a certain turn; with intersection strategy it is similarly determined by the newest device. In either case, without aging the situation in this example does not improve over time.

An improvement needing a little extra memory is to have secondary set C that is like B. A device will gather all the random contributions in C just like in B. When it should clear B, it will instead set $B = C$ and then clear C (and add r). This clearing should occur at certain time intervals. This "sliding window" approach will guarantee that old random seeds will disappear while retaining some memory at the clearing phase. We have used this optimization in our simulations.

The aging of addresses is the reason why a device must not add elements of other L-lists into its own, or why it simply cannot set $B = B \cup B_1 \cup \ldots \cup B_k$. If a device would either one of these, random contributions may remain in the network after devices leave, even if the abovementioned aging mechanisms were used.

V. Device Algorithm

We summarize the algorithm discussion by presenting a "pseudocode" for a program implementing the beacon generation in a node A. We assume that the current state is defined by the node having received n contributions (r_i, L_i, B_i), $i = 1, \ldots, n$, and that we periodically repeat the following algorithm:

1) Set $S = B$
2) For all new received contributions $i = 1, \ldots, n$
   a) Add each r_i to sets B and C.
   b) Add all elements of lists L_i to sets B and C.
   c) Update S with data from B_i based on the update strategy
   d) $i = i + 1$
3) Select some $r_{i_1}, \ldots, r_{i_l}$ and set $L \leftarrow (r_{i_1}, \ldots, r_{i_l})$. This is only a speed-up for diffusion, and the r values may be from advertisments collected during this round, or even come from a cache of older values.
4) If enough time has passed since the last clearing, or if the Hamming weight of B has reached a pre-defined threshold $H(B) > T$, then set $B \leftarrow C$ and $C \leftarrow \{r\}$.
5) Calculate s from S.
6) Advertise (r, L, B) on the next round.

VI. Security Analysis

In the intended application for Wibree location privacy, the following threats against anonymity can be identified:

- *Total break:* Adversary learns the identity root key.
- *Address Tracking:* Adversary can relate two anonymous addresses generated using the same identity root key.
- *Contribution Tracking:* Adversary can track a device based on the random numbers the device is transmitting.
- *Privacy impersonation:* Adversary can replay earlier seen address advertisements, and induce a connection attempt from a device with the knowledge of the corresponding identity root key.

We want to analyze whether any of these threats become stronger and more feasible for the attacker with the use of joint random beacon. For the Wibree location privacy system we thus assume that the devices would not use the random seed r directly in their private address generation $(r, E_k(r))$. Instead all devices would broadcast their respective seeds and, at the same time, combine all received random seeds $r_1, r_2, \ldots, r_m$ to a joint random beacon s, and only then compute the private address as $(s, E_k(s))$ using the identity root keys.

Note that we do not assume that the random numbers are authenticated or identified to belong to devices of a predefined group. In this sense, the device is harvesting randomness from the environment rather than generating it locally. Also note that tracking of a device based on the private address without knowledge of the identity root key is possible only if the adversary can force the device to use a random number which is related to a previously used beacon value. This can be prevented if the two security requirements given in Section II are satisfied.

The second security requirement given in Section II affects the selection of the algorithm used to combine the contributions to the beacon. An adversary can force a new beacon to take on a previously used value if it can contribute with values that makes the new Bloom filter equal to a previous one with large probability. The adversary can use two strategies to achieve this. First it can try to fill the Bloom filter, so that its entropy becomes very small and the beacon has only very few possible values. This attack is prevented by setting an upperbound to the Hamming weight of the Bloom filter. A suitable upperbound is some number between $\frac{1}{2}m$ and $\frac{3}{4}m$, where m is the length of the Bloom filter, as the expected Hamming weight of the Bloom filter filled with n elements is $\frac{1}{2}m$. The second approach is to try to make the Bloom filter S equal to some previously used S. This attack approach is very unlikely to succeed, as the honest devices have contributed to the new Bloom filter with fresh values. The probability that a fresh value belongs to a previous Bloom filter is equal to the probability of false positives.

The joint randomness scenario may also have some effect on the third threat. In case of a loosely connected network, devices may be required to repeat their contributions to the

beacon. Therefore, a device must update its seed regularly, and always when it changes location. The fourth threat present in the basic Wibree privacy solution can be eliminated if the connecting party validates that the advertising party uses the commonly formed beacon determined by the current network neighborhood.

In the application of the group session key establishment the ultimate threat is that the group is forced to use a previously used session key. This can be prevented if the second requirement is satisfied. The first security requirement is not necessary, particularly in scenarios when other honest devices are known to be active in the random beacon generation for the group. Such a feature can be a big advantage in a heterogeneous group, since it allows an energy constrained low-end device to participate in a group session key exchange by just listening to the beacon and without transmitting anything by itself.

In session key generation, the uniqueness of the computed random value for all devices belonging to the same group is of utmost importance. However, in the location privacy application achieving a unique beacon value among the neighboring devices is not absolutely necessary, unless it is necessary to achieve protection against the fourth threat. Accepting a small set of parallel beacon values, e.g. over time, speeds up connectivity, as only new devices need wait for a beacon to form with their own contribution included. Also, computational savings are achieved as soon as the number of different beacon values is reduced from the case where every device is using its own. Then a local address list (may also be implemented as a Bloom filter) can be used to match against many private addresses in parallel, given that the beacon is the same, or from a small set.

VII. Simulations

We have made network simulations intending to illustrate the performance of the algorithm in an ad-hoc network environment. The L-list selection criteria is random choice, p defines the probability that device X hears device Y within the discrete time interval (every device broadcasts once during the interval). In all graphs the strategy $S = B$ is represented by a dashed line, the intersection strategy by a dotted line and the union strategy by a solid line.

The graphs show the number of unique seed values after each time interval (as experienced by the participating nodes) as well as the probability that two randomly selected devices have different seeds. All graphs are aggregates (means) of four simulation runs.

The first figures, Fig. 2 and 3, show the initial beacon convergence with 20 devices in the network, no change in participants and $p = 0.2$. The black lines use $l = 3$ and the gray lines use $l = 0$. The figures illustrate the effect of the L-lists, and the superiority of the union strategy. We can also relate to a typical ad-hoc radio like Wibree in that a reasonable active radio beacon inteval of 30ms would cause our random beacon to be unique and shared by all devices in as little as 200-400 ms.

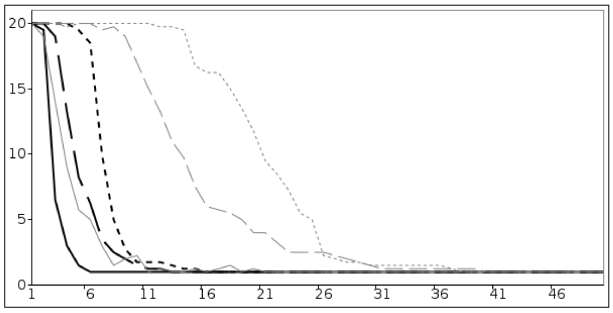

Fig. 2. Number of different beacons.

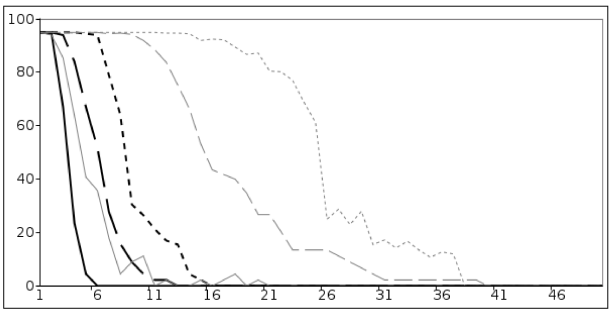

Fig. 3. Probability of two devices having different beacons (percent).

The next scenario shows the dynamic behaviour of the beacon generation as devices enter and leave the network. We have initially 10 devices, $p = 0.3$, $l = 3$ and devices clear their B-sets every 20 time intervals. One, three, five and one device(s) enter at time moments 20, 40, 60 and 80 respectively, and one, three, five and one randomly selected device leave at time moments 90, 130, 170 and 210. As indicated by figures 4 and 5, the variation causes disturbances with only little delay. The random seed of a disappearing device is cleared from the collective C-sets after 0 to 20 rounds and from the B-sets after 20 to 40 rounds. During this time the devices will move gradually from the old seed to a new one.

The last simulation illustrates colliding networks. Here $p = 0.2$, $l = 3$ and B-sets are cleared every 20 turns. At first the network has 10 devices. At time moment 40 a network of 10 devices (with an already established, common beacon of its own) enters the first network. Similarly, at time moments 80

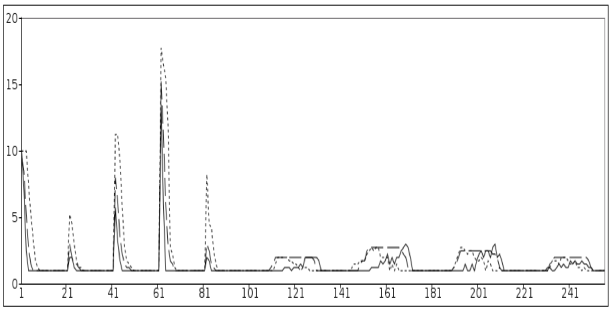

Fig. 4. Number of different beacons.

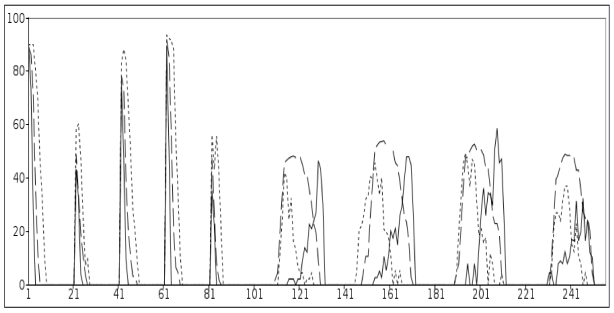

Fig. 5. Probability of two devices having different beacons (percent).

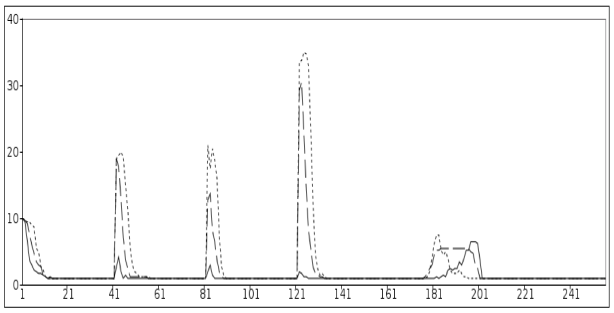

Fig. 6. Number of different beacons.

and 120 two other networks of 5 and 10 devices enter. At time moment 160 ten randomly selected devices leave. See figures 6 and 7. Here the union strategy works well.

We have also conducted preliminary tests of common beacon formation in a network model where the advertisement probability is linearly dependent on the simulated distance between two devices. The results are similar to the results reported above.

VIII. Applicability

We took the the Wibree radio as an example deployment for the ideas presented above, since it includes a privacy addressing scheme that can take advantage of a shared nonce. For a sensor radio, cost in all forms, i.e., transmission, storage and computation should be minimized. We note that the algorithm deployment in a single device as described in section V is short, and relies on no complex cryptographic primitives. The memory requirements needed for the beacon generation is defined by the Bloom filters B and C as well as s, S and r - algorithm steps 2a) and 2c) can be done as advertisements are received, thus no persistent memory is needed to store beacon information (other than the actual private address) from received advertisements.

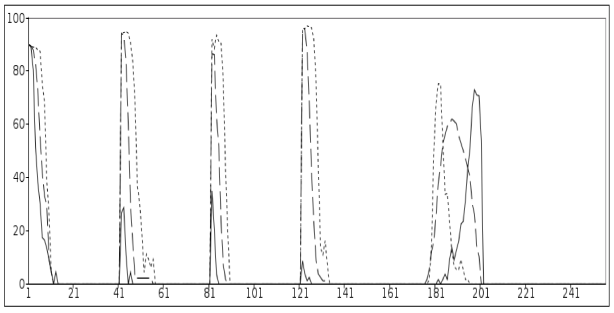

Fig. 7. Probability of two devices having different beacons (percent).

The Wibree radio can advertise at intervals as short as 5ms. But even with less energy-consuming advertisement frequencies the beacon system in ad-hoc network sizes with tens of devices quickly agrees on a common nonce. Simulations also show that the algorithm handles devices entering and leaving the network with only little disturbance in the nonce (in the time domain).

The system is also advantageous in the sense that Wibree specifies a payload option (up to 32 bytes) for advertisements, where the additional beacon information can be included. Unfortunately, the sensor radio cannot broadcast and listen to the network simultaneously - the device must alternate between the two modes. However, as the simulations only assume a 0.2 advertisement throughput, the results should be consistent with the alternating approach.

IX. Conclusions and further work

In this paper we have shown that it is feasible to construct shared nonces in an efficient manner over an ad-hoc radio by device contribution alone. Further work includes simulations with more realistic network models, and strategies, as well as security models for beacon formation in larger networks where it can be assumed that not all advertising devices need to contribute to the shared nonce.

References

[1] Wibree home page, "http://www.wibree.com."

[2] U. Maurer, "Conditionally-perfect secrecy and a provably-secure randomized cipher," *Journal of Cryptology*, vol. 5, no. 1, pp. 53–66, 1992.

[3] R. Gilbert, K. Johnson, S. Wu, B. Y. Zhao, and H. Zheng, "Location independent compact routing for wireless networks," in *MobiShare '06: Proceedings of the 1st international workshop on Decentralized resource sharing in mobile computing and networking*. New York, NY, USA: ACM, 2006, pp. 57–59.

[4] P. Hebden and A. Pearce, "Data-centric routing using bloom filters in wireless sensor networks," *Intelligent Sensing and Information Processing, 2006. ICISIP 2006. Fourth International Conference on*, pp. 72–77, Oct. 15 2006-Dec. 18 2006.

[5] J.-H. Son, H. Luo, and S.-W. Seo, "Authenticated flooding in large-scale sensor networks," *Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on*, pp. 8 pp.–, 7-10 Nov. 2005.

[6] M. Gruteser and D. Grunwald, "Enhancing location privacy in wireless LAN through disposable interface identifiers: A quantitative analysis," *Mobile Networks and Applications*, vol. 10, pp. 315–325, 2005. [Online]. Available: http://www.springerlink.com/content/k810777376g06432/

[7] J. Lindqvist and L. Takkinen, "Privacy management for secure mobility," in *WPES '06: Proceedings of the 5th ACM workshop on Privacy in electronic society*. New York, NY, USA: ACM Press, 2006, pp. 63–66.

[8] F.-L. Wong and F. Stajano, "Location privacy in Bluetooth," in *Security and privacy in ad-hoc and sensor networks*, ser. Lecture Notes in Computer Science. Springer-Verlag, 2005, pp. 176–188.

[9] C. Gehrmann and K. Nyberg, "Enhancements to Bluetooth baseband security," in *Proceedings of NordSec 2001*, 2001.

[10] J.-E. Ekberg, "Implementing address privacy," in *Ubicomp 2007 Workshop Proceedings: IWSSI 2007*, 2007, pp. 481–485.