Institut EURECOM
Corporate Communications
2229, route des Crêtes BP 193
06904 Sophia Antipolis
(France)

Research Report[a] N^o 76 — RR-03-076

Background Signature for Sensor Networks

Laurent Bussard and Yves Roudier

June 18, 2003

[a]Eurecoms research is partially supported by its industrial partners: Bouygues Telecom, Fondation d'entreprise Groupe Cegetel, Fondation Hasler, France Telecom, Hitachi, ST Microelectronics, Swisscom, Texas Instruments, and Thales

Background Signature for Sensor Networks

Laurent Bussard and *Yves Roudier*
Institut Eurécom
Corporate Communications
2229, route des Crêtes BP 193
06904 Sophia Antipolis
(France)
{bussard, roudier}@eurecom.fr

June 19, 2003

Abstract

Sensor networks are characterized by lack of trusted infrastructure and severe hardware limitations in terms of computational power and memory size. Securing the exchanges between a set of distributed sensors is challenging. On one hand, ad hoc infrastructures imply a lack of a priori trust between entities that can dynamically appear and disappear. In this context, it is necessary to ensure non-repudiation of origin and integrity of alarm messages sent by sensors: asymmetric cryptography must be available. On the other hand, asymmetric cryptography is too costly and drastically increases the response time of sensors. For instance, due to the signature slowdown, sending a measure or an alarm can take several seconds. This paper presents a new approach that allows strong acceleration of signatures by pre-computing the major part of the signature process. This scheme is based on one-time signatures and pre-computation of RSA signatures. This paper studies the impact of this scheme on computation and response time when a sensor has to send a signed message. Memory and communication restrictions are taken into account to find an optimal structure of one-time key pairs.

1 Introduction

Pervasive computing, ad-hoc networks and sensor networks are exploding in popularity because they are not only a vision but are becoming a reality. Pervasive computing assumes that smaller computers are increasingly spreading in our environment. Being available ubiquitously in the devices and appliances that we use everyday, this embedded computing power increases their capacities. Furthermore, these computers are also increasingly interconnected into varying range networks, thus enabling access to a pervasive flow of information and services. Ad-hoc networking assumes that there is no infrastructure and that each involved device has to provide services from network level (e.g. routing packets) to application level (e.g. displaying data). Sensor networks are more specific in that devices

(sensors) have a very small computational power: sensors are small and cheap chips that are spread in the environment to provide accurate measures (e.g. light, heat, and pressure).

Enabling security features in ad-hoc sensor networks is challenging because of the lack of infrastructure, the lack of a priori trust among involved entities and, the lack of computational power. When sensors send messages that can be measures or alarms, message integrity and non-repudiation of origin are necessary: signature of messages is necessary. It is also necessary to ensure message authentication but common authentication, which is based on identities, is meaningless in such an open environment. It is the reason why certification of sensors is proposed to strongly linking attributes to sensors. For instance, it is possible to verify that a message was sent by a sensor with a given attribute (e.g. belonging to a given company).

A way to let sensors sign message is required. This signature has to be very efficient so that a sensor can send a signed message immediately when triggered by an external event and the overall computation cost has to be relatively small to fit computational limitations of sensors and/or to limit their energy consumption.

We first give a description of the environment and security requirements. Section 3 shows how the combination of one-time signatures and pre-computed RSA signatures can speed the response time of a sensor up. Section 4 describes other work aiming at accelerating the computation of signatures. Possible enhancement of the basic scheme are presented in Section 5: an optimal tree of one-time key pairs and a way to pre-distribute part of the data are proposed. Finally, Section 6 shows how this signature scheme can be used for ensuring non-repudiation of origin and for certifying sensors.

2 Problem Statement

A set of sensors $S_1 \cdots S_p$ is deployed in an environment. Each sensor is in charge of sending some alert message when an event occurs (temperature increase, proximity detection, etc.). A set of observers $O_1 \cdots O_q$ collects messages sent by sensors and analyzes those messages. There is no *a priori* trust between sensors and observers that can belong to different domains. There is no communication infrastructure (*ad hoc*): sensors and observers can appear and disappear in the environment (see Figure 1).

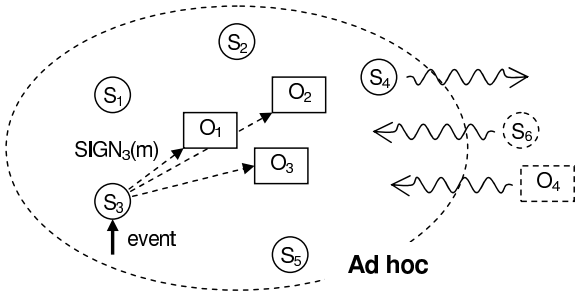

Figure 1: Overview of the ad hoc environment: sensors and observers

Sensors offer a low computation power and cannot afford crypto-processors. They only have a small memory and relatively low speed communication channels (wired or wireless). Finally, sensors need a low consumption rate but are regularly powered (solar power, kinetic energy, etc.).

The challenge tackled by this paper is to ensure non-repudiation and certification (i.e. signature) in such a context with fast enough response time when an event triggering an alert message occurs.

3 Pre-computing Signatures

3.1 Basic Principle

Sensors are triggered by external events such as temperature increase, movement, etc. and have to react quickly by sending a signed message. In order to shorten the response-time, this paper proposes a scheme based on one-time signatures (OTS) [8], which can be computed quickly but are only usable once. Figure 2 presents this approach: *Before being triggered*, the sensor creates a pair of one-time public and private keys (OTPK and OTSK) and signs the one-time public key with its RSA private key. *After being triggered*, the sensor signs in an efficient way the message with the one-time private key. Next, the sensor sends the one-time signature of the message and the RSA signature of the one-time public key.

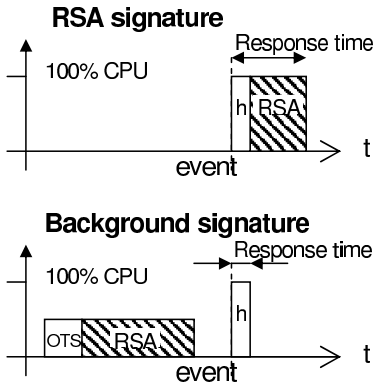

Figure 2: Principle of background signatures

Merkle [8] proposed an efficient one-time signature construction: the signer generates a one-time private key, which is a vector OTSK = $\{R_1, \ldots, R_n\}$ of $n = l + \lfloor 1 + \log_2(l) \rfloor$ random numbers where l is the digest size of the message to sign. The one-time public key is OTPK = $\{\mathrm{h}(R_1), \ldots, \mathrm{h}(R_n)\}$.

When the signer wants to sign a message m, he gets the l bit digest $h(m)$, counts the number of bits set to '0' in this digest and appends this checksum to $\mathrm{h}(m)$ so that he obtains a n bit digest d. This checksum is necessary to ensure the integrity of the signed message. Indeed, the one-time signature scheme allows pretending that bits set to '1' are set to '0'. The checksum avoids this attack against the one-time signature scheme. The signature $\mathrm{SIGN}_{\mathrm{OTSK}}(m)$ is:

$$\text{if } d[i] = 1 \ : \text{ release } R_i \quad 1 \leq i \leq n \quad \text{(see Table 1)}$$

For instance, if $\mathrm{h}(m) = 10110$ (thus $l = 5$ and $n = 8$), there are two bits set to '0' and the checksum is 010 then $d = 10110010$ and the signature $\mathrm{SIGN}_{\mathrm{OTSK}}(m) = \{R_1, R_3, R_4, R_7\}$. The verifier can check that the received R_i is a signature done with the private key corresponding to the one-time public key OTPK:

$$h(R_i) \stackrel{?}{=} \mathrm{OTPK}[i] \quad \forall i \mid R_i \in \mathrm{SIGN}_{\mathrm{OTSK}}(m)$$

We define *background signature* (BS) as the combination of pre-computed asymmetric signatures and one-time signatures based on Merkle's scheme. Before being triggered, the sensor generates n random numbers (one-time private key), computes n hash functions (one-time public key) and one RSA signature. As shown in Figure 2, when the sensor is triggered, it computes one hash function to get the digest of the message to sign. The one-time signature does not require more security function.

Background signatures allow to accelerate reactions to an event (see 3.2) but the overall cost of a signature slightly increases in term of size and computational time (see 3.3):

- *RSA*: m, $\mathrm{SIGN}_{\mathrm{SK}}(m)$

- *BS*: m, $\mathrm{SIGN}_{\mathrm{OTSK}}(m)$, OTPK, $\mathrm{SIGN}_{\mathrm{SK}}(OTPK)$

However, Section 5 shows how the overall cost can be reduced when one RSA signature is used for multiple one-time signatures.

PK_A	public key of entity A
SK_A	private key of entity A
$m[i]$	i^{th} bit of message m
$m_1 \| m_2$	concatenation of messages m_1 and m_2
$\mathrm{E}_{\mathrm{K}}(m)$	plaintext m encrypted with key K
$\mathrm{h}(m)$	digest of data m with hash function h
$\mathrm{SIGN}_{\mathrm{SK}}(m)$	m signed with the private key SK: $\mathrm{SIGN}_{\mathrm{SK}}(m) = \mathrm{E}_{\mathrm{SK}}(\mathrm{h}(m))$.
OTPK	one-time public key
OTSK	one-time private key
$\mathrm{SIGN}_{\mathrm{OTSK}}(m)$	one-time signature of plaintext m (see section 3.1)

Table 1: Cryptographic notations used in this paper.

3.2 Impact on Response Time

The background signature scheme allows an important speed-up factor. When the sensor is triggered by an event, it computes the digest of the message and releases an average of $1/2 \cdot (l + \log_2(l))$ pre-computed private factors R_i.

$$\text{speed-up factor} = \frac{\text{Response time of RSA sig}}{\text{Response time of BS}}$$

Figure 3 shows the speed-up factor as a function of the message size for different key lengths. Those value have been measured on a Pentium 4 running at 2 GHz with a Java program based on the cryptographic library Bouncy Castle [6]. Section 3.4 shows that the speed-up ratio is almost independent of the chosen platform even if the response times can strongly vary. When a 2 kBytes message is signed with RSA (1024 bits), the speed-up factor is about 300. In other words, a slow sensor that requires 3 seconds to sign a message can react in 10 milliseconds when background signature is used. The impact of the communication bandwidth will be described subsequently.

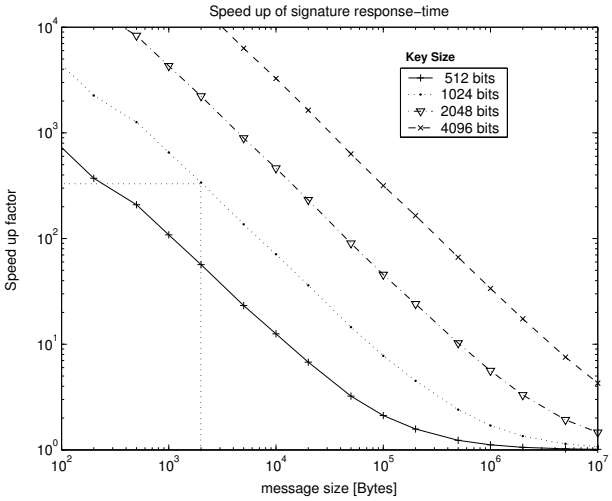

Figure 3: Impact of background signatures on response time with different key sizes

3.3 Impact on Computation

The basic background signature scheme adds overhead: the pre-computation of a signature is more expensive than its computation. Indeed, in both cases a RSA signature has to be computed and the former requires the generation of a one-time key pair. Left of Figure 4 shows the impact

on the signature. For instance, a 1024 bit RSA signature of a 2 kBytes message is 6% more expensive when background signature is required. The impact on signature verification is more important at 130% of overhead for the same message. However, it is assumed that signature verification is only performed by powerful observers.

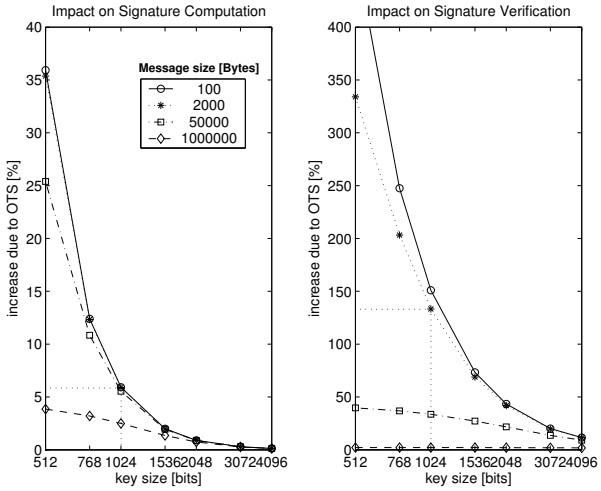

Figure 4: Impact of background signatures on computation with different message size

Section 5 shows how it is possible to reduce the cost of background signatures by using one asymmetric operation to sign a set of one-time public keys.

3.4 Platform Dependences

For the sake of simplicity and efficiency, measures presented in this paper have been done on a workstation. However, the ratio between signatures and digest is relatively independent of the memory and computational power. Table 2 compares some software approaches: measures on a workstation (Pentium 4 with J2SE 1.4 and Bouncy Castle), measures on a PDA (iPaq 200 MHz with Jeode and Bouncy Castle), and estimation based on results of an experimental smart card [3] (C implementation, RISC 25 MHz). When a crypto-processor is used, this ratio can decrease down to 500 but it is still interesting to use background signatures.

Platform	**RSA-Sig** [s]	**Hash** [s]	**Ratio**
Workstation	$19 \cdot 10^{-3}$	$5 \cdot 10^{-6}$	3800
PDA	$190 \cdot 10^{-3}$	$120 \cdot 10^{-6}$	1600
Smart card	$490 \cdot 10^{-3}$	$350 \cdot 10^{-6}$	1400

Table 2: Ratio between signature (RSA 1024) and digest (SHA-1 of 512 bit message) on different platforms

3.5 Impact of Event Occurrence

Figure 5 shows the result of a simulation based on the previous measures. The response time is a function of the frequency of event occurrence. RSA is compared with different types of background signatures. The message size is 2 kBytes. Top and bottom plots respectively use 512 bit and 1024 bit keys. The following items describe all simulation of Figure 5:

- *RSA*: a whole RSA signature is done each time an event occurs. With a 512 bit RSA key, the signature lasts 3 ms on the workstation and thus, when the frequency of events is bigger than 330 signature per second, the response time grows exponentially (see top of Figure 5). With RSA 1024, a signature lasts 19 ms and thus the frequency of events should not be more than 52 signature per second (see bottom of Figure 5).

- *Basic BS*: the basic background signature

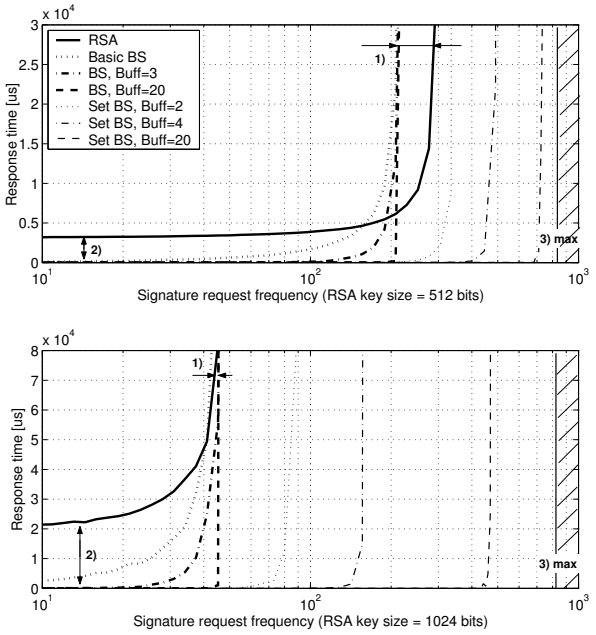

Figure 5: Impact of signature request frequency [sig/s] on response time

scheme is the one described previously. The sensor prepares one pair of one-time keys and signs the one-time public key with its RSA private key. When an event occurs, the one-time private key is used then a new one-time key pair is generated and signed. Section 3.3 shows that the creation of the one-time key pair adds some overhead that limits the frequency of event occurrence.

- *BS, Buff 3*: background signature with signature buffer of size 3. The sensor can store up to three one-time key pairs in a buffer. The sensor prepares pairs of one-time keys and signs each one-time public key. When an event occurs, a one-time private key is used and a new one-time key pair can be generated and signed.

- *BS, Buff 20*: background signature with signature buffer of size 20. The sensor can store up to twenty one-time key pairs in a buffer.

- *Set BS, Buff 2*: background set of signature with signature buffer of size 2. The sensor creates two one-time key pairs and uses only one RSA signature to sign both of them. This scheme has a stronger impact when the key size is large because the maximum event frequency (i.e. when the buffer is infinite) does not depend on the key size.

- *Set BS, Buff 4*: background set of signature with signature buffer of size 4. The sensor creates four one-time key pairs and uses only one RSA signature to sign all of them.

- *Set BS, Buff 20*: background set of signature with signature buffer of size 20. The sensor creates eight one-time key pairs and uses only one RSA signature to sign all of them. The effect of a large buffer is negligible with small key sizes (RSA 512). The computation cost of a background signature based on RSA 1024 with 20 memory cells is smaller than RSA 512 with 4 memory cells.

Label 1) of Figure 5 shows the acceleration factor due to the background signature scheme (see Figure 3). Label 2) shows the computation overhead of background signatures (see Figure 4).

When a set of one-time public keys is signed in one RSA operation, the response time does not change but the overall background computation t_{BS} is reduced.

$$t_{BS} = t_{OTS} + \frac{t_{RSA}}{b} + t_{hash}$$

Where t_{OTS}, t_{RSA}, and t_{hash}, are the durations required to compute respectively $l + \log_2(l)$ random numbers and their digests, a RSA signature, and the digest of the message to sign. b is the buffer size. The maximum frequency occurrence of events is:

$$\lim_{b \to \infty} t_{BS} = t_{OTS} + t_{hash}$$

In Figure 5, the $\lim_{b\to\infty} t_{BS} = 1.2$ ms. Label 3 is the corresponding maximum frequency of event occurence: 833 signatures per second. To summarize, the response time does not depend on the RSA key size and, when there is enough memory ($b \to \infty$), the background computation and maximum frequency of event occurrence do not depend on key size.

4 Related Work

This section presents different approaches that have been proposed to accelerate the computation of a digital signature.

Server-Aided Signature: a small tamper-resistant trusted module uses the computational power of an untrusted server to compute a signature. The server can try to obtain the secret (i.e. private key) of the module or to cheat with a false result. The trusted module must protect its secret and verify the computation received from the server. [2] proposes a solution enabling server-aided RSA signatures. With enough bandwidth between the server and the trusted module, this scheme can be ten times faster than the trusted module alone. The drawback of this approach is that it implies the availability of a powerful server each time a signature has to be computed. Finally, numerous former server-aided signature schemes have been found insecure [9].

Verifiable Server: an approach similar to server-aided signature is proposed in [1]. A trusted module asks a trusted server to sign some data. The server is in charge of all asymmetric cryptography operations. However, the signature scheme is modified so that a valid signature cannot be generated without the help of the security module. Non-repudiation is ensured even if the private key is hold by the server. As in the previous scheme this approach suffers from relying on trusted servers.

Batch Signatures: the idea of batch signatures [10] is to do only one asymmetric operation for a set of signatures. A set of messages are linked and signed together. For instance, a hash tree can be used to ensure that the signature of a message can be verified without knowing the other messages. This approach is useful when a set of messages has to be signed simultaneously but it cannot be used to accelerate the response-time of individual signatures.

Other Public Key Cryptosystems: it is also possible to replace RSA by another signature scheme. For instance, the McEliece cryptosystem [7] can be used to efficiently sign data [4], the main drawback of this approach being the size of the public key. Elliptic curve cryptography could be investigated too. However, all asymmetric cryptosystem are computationally expensive compared with hash functions.

5 An Enhancement: Signature Trees

This section studies how one RSA signature can be used to enhance BS for a set of messages. It is not similar to batch signatures [10] because in background signature, there are no simultaneous messages to be signed. The principle is first de-

scribed and an optimal structure of one-time key pairs is proposed.

5.1 Principle

Using a tree of one-time signatures is not a new idea: in [8], binary trees were proposed. However, constraints due to sensor networks suggest new tree structures.

A complete k-ary tree of height h has n nodes and can be used to sign $n-1$ messages.

$$n = 1 + k + k^2 + \cdots + k^h = \sum_{i=0}^{h} k^i = \frac{k^{h+1} - 1}{k - 1}$$

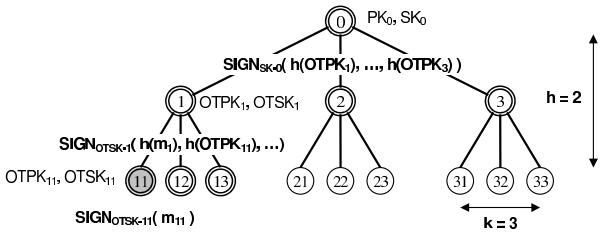

Figure 6: Tree of one-time signatures

The root key pair (SK_0, PK_0) does not change since it is the RSA key pair of the sensor. A set of $n-1$ nodes (i.e. one-time key pairs) is created and the RSA private key of the sensor is used to sign the public keys of depth one. One-time signatures of messages can then start, signatures being done alphabetically (i.e. 1, 11, 111, 112, 113, 12, 121, etc.). When message m_1 is signed with private key $OTSK_1$, this message is signed together with the public keys of the next level in the tree: $\mathrm{SIGN}_{\mathrm{OTSK}_1}(\mathrm{h}(m_1)\|\mathrm{h}(OTPK_{11})\|\cdots)$. The result is a chain of signatures linking the root RSA key PK_0 to a message (e.g. m_{11}):

- $\mathrm{SIGN}_{\mathrm{SK}_0}(\mathrm{h}(OTPK_1)\|\cdots\|\mathrm{h}(OTPK_3))$
- $\mathrm{SIGN}_{\mathrm{OTSK}_1}(\mathrm{h}(m_1)\|\mathrm{h}(OTPK_{11})\|\cdots\|\mathrm{h}(OTPK_{13}))$
- $\mathrm{SIGN}_{\mathrm{OTSK}_{11}}(\mathrm{h}(m_{11}))$

An observer receiving message m_{11} needs the following data to verify the signature:

- m_{11}
- $\mathrm{SIGN}_{\mathrm{OTSK}_{11}}(\mathrm{h}(m_{11}))$
- $OTPK_{11}$
- $\{\mathrm{h}(m_1), \mathrm{h}(OTPK_{12}), \cdots, \mathrm{h}(OTPK_{13})\}$
- $\mathrm{SIGN}_{\mathrm{OTSK}_1}(\mathrm{h}(m_1)\|\mathrm{h}(OTPK_{11})\|\cdots\|\mathrm{h}(OTPK_{13}))$
- $OTPK_1$
- $\{\mathrm{h}(OTPK_2), \cdots, \mathrm{h}(OTPK_3)\}$
- $\mathrm{SIGN}_{\mathrm{SK}_0}(\mathrm{h}(OTPK_1)\|\cdots\|\mathrm{h}(OTPK_3))$
- PK_0

Part of this information can be pre-installed in the observer (e.g. PK_0) or pre-distributed with the precedent message (e.g. $OTPK_1$, etc.), the rest being sent with the signature.

5.2 Optimal Tree

Two parameters have to be taken into account to find an optimal tree: on one hand, the main goal is to limit the size of the messages sent to have a short response time. On the other hand, it is necessary to limit the computation: memory has to be well managed in order to associate a large number of one-time key pairs with one RSA signature.

Message size is directly related to the height of the tree. Each new level increases the signature length with the following information:

- $OTPK_{z1}$
- $\{\mathrm{h}(m_z), \mathrm{h}(OTPK_{z2}), \cdots, \mathrm{h}(OTPK_{zk})\}$
- $\mathrm{SIGN}_{\mathrm{OTSK}_z}(\mathrm{h}(m_z)\|\mathrm{h}(OTPK_{z1})\|\cdots\|\mathrm{h}(OTPK_{zk}))$

The resulting message overhead is:

$$O_h = h \cdot \left(l \cdot k + \frac{3}{2} \cdot l \cdot (l + \log_2(l)) \right)$$

The shortest messages are sent when the tree height h is defined as follows: $O_{h-1} > O_h <$

O_{h+1}. With $k \cong \sqrt[h]{n}$, it is necessary to choose h so that:

$$(h-1)\sqrt[h-1]{n} - h\sqrt[h]{n} > \frac{3}{2}(l + \log_2(l))$$

and

$$h\sqrt[h]{n} - (h+1)\sqrt[h+1]{n} < \frac{3}{2}(l + \log_2(l))$$

Figure 7 shows the message size overhead for different tree heights (SHA-1 being used, $l = 160$ bits). This figure shows that the minimum message size is obtained when $h = 1$ if $n - 1 < 281$, when $h = 2$ if $281 < n - 1 < 29438$, etc.

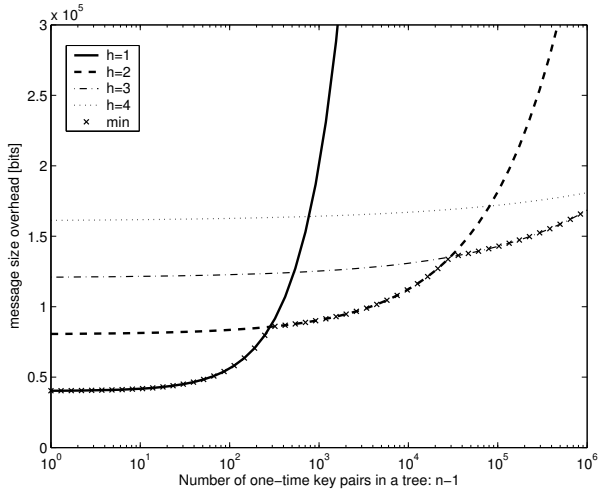

Figure 7: Impact of tree's depth on message size

When a message is signed, part of the one-time key pairs has to be defined. All keys linking the current key to the RSA key are required. In Figure 6, key pairs 0, 1, and 11 are needed to sign message m_{11}. Moreover, each key pair signed by those keys is defined: key pairs 2, 3, 12, and 13 are defined and will be used subsequently. In other words, $h \cdot k$ key pairs are stored. It is possible to show that 3-ary trees maximize the number of nodes (i.e. one-time signatures per RSA signature) with a fixed amount of memory. Response time and communication restrictions advocate for a limited height (i.e. $h = 1$ or $h = 2$ when $n < 10^4$) and the memory is well used when 3-ary trees are chosen (i.e. $k = 3$). Fast response time being our main goal, we have decided to use $h = 2$. In this setting, the best tree from a memory point of view has the structure proposed in Figure 8. The value under each node is the number of one-time key pairs that have to be stored when the one-time private key is used to sign a message. The tree is asymmetric because part of the one-time key pairs that have been used previously can be erased; i.e. only the hash of the one-time public key is necessary. For instance, when key 3 of Figure 8 is used, it is necessary to store eight one-time key pairs (3,4,5,31,32,33,34,35), when key 31 is used, the same key pairs have to kept but when key 34 is used, only five key pairs have to be stored (3,4,5,34,35). In this example, by maintaining eight key pairs at any time, it is possible to compute thirty signatures.

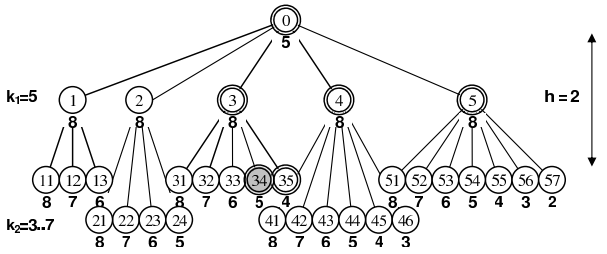

Figure 8: Optimal key pair tree

The memory used is $m = \lceil k_1 \cdot 3/2 \rceil$ and the number of possible signatures is $n - 1 = k_1 + k_1^2$.

To optimize the communications, it is possible

to take advantage of the tree structure in order to pre-distribute part of the data necessary to verify the next signature. In Figure 9, $OTPK_{12}$, etc. have been sent with the previous message signature. Observers that received the previous signature can verify the new message more efficiently than arriving observers. Due to the tree structure, redundant communication is limited.

6 Discussion

This section presents some applications of background signature and possible enhancements.

6.1 Non Repudiation of Origin

When there is no a priori trust relationship between sensors and observers, it is important to offer non-repudiation of origin so that observer can prove subsequently that a given sensor sent a message. Imagine for instance that an observer automatically calls the firemen when some surrounding sensors indicate a sudden temperature increase. If there is no fire, it is important that the observer have a way to prove that it received some measures before acting.

Letting a sensor and an observer share a secret is not sufficient for non-repudiation. Indeed, a message authentication code allows the verification of message integrity and the authentication of the sender but it cannot be used as a proof in case of a dispute between the sensors' owner and the observer. Signature is necessary for ensuring non-repudiation of origin and background signature allows signature in sensor networks.

6.2 Certifying Sensors

Signatures are useless if the signer is not known. Because identity of signers (e.g. serial number of sensors) is meaningless in open context, it is interesting to certify attributes of sensors. For instance, the simple public key infrastructure (SPKI) [5] can be used. The manufacturer or some local certification authority (CA) can certify that a sensor has some properties, that it belongs to a given entity, or when it has been controlled. Those data being linked to the sensor's public key, an observer that receives a message from a unknown sensor has the following information: alarm message m was sent by a sensor that has been deployed by a partner and that has been controlled two months ago. Background signature allows sending this certified message quickly.

6.3 Avoiding Replay Attacks

The replay attack is a well-known attack in which an attacker sends an old message to fool the receiver. Generally, nonces or some representation of time (e.g. counter, time stamps) are used to avoid that a message be replayed. Assuming that sensors cannot afford a real-time clock, a simple way to avoid replay attacks when messages are signed by sensors is to have an observer in charge of periodically sending time stamps that can be used to increment some counter acting as a static clock in each sensor.

6.4 Letting Sensors Verify Signatures

This paper proposes a way to shorten the response time of sensors. This scheme is adapted to sensors working in *push mode*, i.e. the sensor immediately sending a signed message with relevant information (e.g. temperature of a container) when an event occurs. When sensors are in *pull mode*, i.e. when observers periodically send requests to sensors in order to get

some measure, the response time is less critical. In this case, it can be important to have a way to authenticate observers. Background verification of signatures could be implemented to 'pre-verify' part of the signature. For instance, a server or another sensor could send a OTPK signed with its RSA private key. Each sensor receiving this data could verify the signature as a background task and wait for a message signed with the related one-time private key. Unfortunately, the response time of the signature verification is longer than the response time of the signature computation because the $(l+\log_2(l))/2$ hash functions cannot be pre-computed.

6.5 Limitations of Background Signature Scheme

The major limitation of the scheme proposed in this paper is the size of a signature (about ten times a RSA 1024 signature). Background signature scheme is efficient when the computational power is low and when the bandwidth is relatively large. For instance, a cell-phone needs about ten seconds to sign a document when asymmetric cryptography (RSA 1024) is done by the device itself (e.g. J2ME) and not by a crypto-processor (e.g. SIM card). If Bluetooth is used (100 kbits/s measured), the speed-up factor is about fifty. However, if the same scheme is used with an iPaq and the same communication channel, there is no more speed-up effect. This scheme only fits devices with restricted computational power which is typical of sensors.

Let us assume a sensor with low computational power that needs 60 seconds to compute a RSA 1024 signature. A tree with height $h = 2$ and $k_1 = 5$ is chosen (Figure 8). It allows 30 signatures and requires memory to store 8 one-time key pairs. The communication channel offers 64 kbits/s. In this case (Figure 9), the response time after an event is 0.3 s when keys are predistributed (200 times faster) and 1.3 s for observers that did not receive the keys in advance (46 times faster).

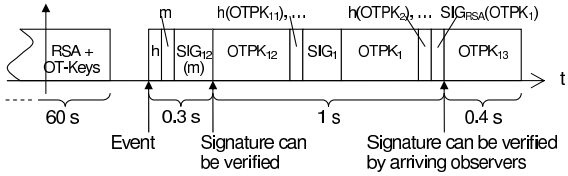

Figure 9: Response time of a sensor using background signatures

7 Conclusion

The background signature scheme proposed in this paper has some interesting properties. *Background signature shortens response time*: this paper has shown that using background signature in sensor networks allows a fast response time even when non-repudiation of origin, message integrity, and sensor certification have to be ensured. The acceleration factor is higher with short messages, long keys, and large bandwidth. *Background signature freely enhances security*: the response time of a sensor does not depend on the key size. Moreover, when there is enough memory to create a large tree of key pairs, the overall computation does not depend on the key size. In this context, using background signature based on RSA 512 or RSA 2048 exhibits the same performances. *Background signature can be seen as a 'battery'*: an important time (and energy) consuming activity of sensors is the signature of messages. When the small battery of an inactive sensor is filled and some power is

available, the sensor can use CPU cycles (that cannot be stored) and available electric power (that can no more be stored) to pre-compute some signatures. Memory is used to store pre-computation and somehow extends the battery capability. When an event occurs, the sensor can react quickly and requires less energy for computation.

The main limitation of this scheme is the increase of message size due to the one-time signature. In some cases, the response time profit in term of computational complexity can be eliminated by the communication overhead. In other words, this scheme is only valuable when the communications are fast or when the computation is slow (see Section 6.5). Fortunately, this assumption is realistic in case of sensor networks. To summarize, this scheme can be used to accelerate signatures done by cell-phones or sensors but is useless in case of smart cards with crypto-processors or powerful workstations.

The generation of new signature trees is still an open topic. When each one-time private key of a tree has been used, it is necessary to compute a new RSA signature. The sensor cannot sign any message during this phase that can last a long time when the key size is big. Two trees could be combined to avoid this problem.

References

[1] K. Bicakci, N. Baykal, *SAOTS: A New Efficient Server Assisted Signature Schema for Pervasive Computing*, In proceedings of Security in Pervasive Computing SPC'03, March 2003.

[2] P. Bégiun, J.J. Quisquater, *Fast Server-Aided RSA Signatures Secure Against Active Attacks*. In proceedings of CRYPTO'95, pages 57-69, 1995.

[3] Cascade project, *Chip Architecture for Smart CArds and portable intelligent DEvices*, European Commission, Esprit Program (EP8670).

[4] N. Courtois, M. Finiasz, N. Sendrier *How to achieve a McEliece-based digital signature scheme*. In Advances in Cryptology - ASIACRYPT 2001, number 2248 in LNCS, pages 157-174. Springer-Verlag, 2001.

[5] C.M. Ellison, B. Frantz, B. Lampson, R. Rivest, B.M. Thomas and T. Ylonen, *SPKI Certificate Theory* RFC 2693, 1999, expired.

[6] Legion of the Bouncy Castle, *Java Crypto APIs*, http://www.bouncycastle.org/

[7] R. McEliece, *A public-key cryptosystem based on algebraic coding theory*. In Jet Propulsion Lab. DSN Progress Report, 1978.

[8] R. Merkle, *A digital signature based on a conventional encryption function*, In proceedings of Crypto'87, 1987

[9] J. Merkle, *Multi-Round Passive Attacks on Server-Aided RSA Protocols*. Proceedings, CCS'00, pp. 102–107, 2000.

[10] C. Pavlovski, C. Boyd, *Efficient Batch Signature Generation using Tree Structures*, International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC'99), City University of Hong Kong Press, pp.70-77, 1999.