(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
3 May 2007 (03.05.2007)

PCT

(10) International Publication Number
WO 2007/048659 A1

(51) International Patent Classification:
G07C 9/00 (2006.01)

(21) International Application Number:
PCT/EP2006/066367

(22) International Filing Date:
14 September 2006 (14.09.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
05300873.6 27 October 2005 (27.10.2005) EP

(71) Applicant *(for all designated States except US)*: **INTERNATIONAL BUSINESS MACHINES CORPORATION** [US/US]; New Orchard Road, Armonk, New York 10504 (US).

(71) Applicant *(for MG only)*: **COMPAGNIE IBM FRANCE** [FR/FR]; Tour Descartes, La Défense 5, 2, Avenue Gambetta, F-92400 Courbevoie (FR).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **BAUCHOT, Frédéric** [FR/FR]; 299 Chemin du Vallon, La Tourraque, F-06640 Saint-Jeannet (FR). **MARMIGERE, Gérard**; Quartier Le Patrimoine, F-06340 Drap (FR). **BERDAH, Maurice** [FR/FR]; 19, Allée du Professeur Dubos, F-95350 Saint-Brice-sous-Forêt (FR).

(74) Agent: **ETORRE, Yves Nicolas**; Le Plan du Bois, F-06610 La Gaude (FR).

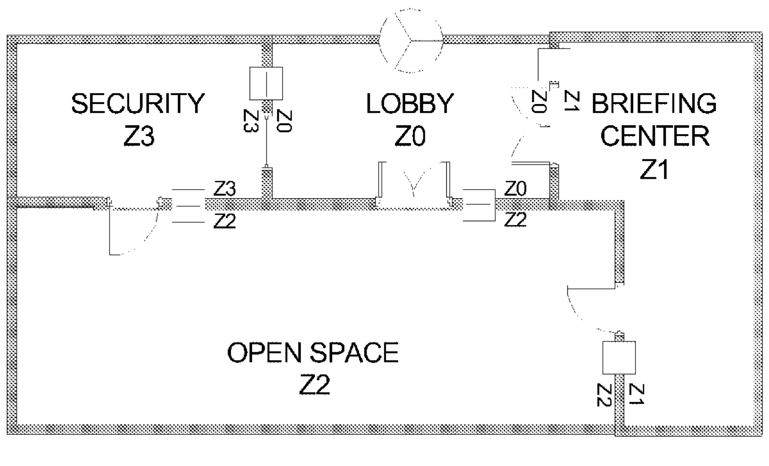
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declaration under Rule 4.17:
— *of inventorship (Rule 4.17(iv))*

[Continued on next page]

(54) Title: SYSTEM AND METHOD FOR DYNAMICALLY MANAGING BADGE ACCESS



(57) Abstract: The present invention discloses methods, systems and computer programs for dynamically managing access to different protected zones with different security levels by means of badges and badge readers, access control being performed both when entering and leaving a protected zone. Each area or zone protected by the method and system according to the present invention is identified by a unique Zone Identifier Z(i). Each zone can be accessed through a Key K(i) hold by a badge and read by a reader. Each zone is associated with a maximum time duration T(i) during which a badge is authorised to stay in the zone. Each badge within a zone Z(i) is identified by an Identifier ID(i). To move from a zone Z(i) to a zone Z(j), a badge with identifier ID(i) must show that it holds the key K(i). If it is the case, the badge receives the key K(j) which allows afterwards to leave the zone Z(j). When a zone Z(i) is empty (no badge present in the zone), the server has the possibility to update the key K(i). The main principles of the present invention are the following: A badge reader can't stay indefinitely within a given zone. Badge readers are not only used to enter a zone, but also to leave a zone. The key used to leave a zone is dynamically passed to the badge when this badge is used to enter in the zone. Key are changed when a zone it empty.

WO 2007/048659 A1

Published:
— *with international search report*

SYSTEM AND METHOD FOR DYNAMICALLY MANAGING BADGE ACCESS

Field of the invention

The present invention relates to security and more particularly to methods, systems and computer programs for dynamically managing access to different areas with different security levels by means of badges and badge readers.

Background of the invention

The problem that the present invention proposes to solve can be illustrated by the following example. Figure 1 represents a building belonging to a private company, with different areas, each of them associated with a specific security level:

- The lobby, with a security level Z0, is a public area where anybody has access to.
- The briefing center, with a security level Z1, is an area of limited security, accessible to the customers of the company. The access is granted for the people holding a badge.
- The open space, with a security level Z2, is an area of high security, only accessible to the employees of the company. The access is granted for the people holding a badge.
- The security center, with a security level Z3, is an area of very high security, only accessible to security staff and authorized company personal. The access is granted for the people holding a badge.

It must here well understood that the building layout does not allow all transitions between the different areas, and hence between the different security levels. With the previous building layout, conventional access techniques define different security levels, according to a given hierarchy, so that a badge can give access either to the level Z1 only, or to the levels Z0 and Z1, or to the levels ZO, Z1 and Z2, or to all the levels Z0 through Z3. With such a scheme, some security breaches are difficult to avoid, as shown with the following examples:

- Any stolen badge granting access to a security level Zi can be used for fraudulently accessing areas with a security level lower than or equal to Zi.

- Extended (and therefore suspicious) stay within a given area can't be easily detected.
- Any attempt to move from security level Z3 to security level Z0 without passing through the security level Z2 can't be detected.
- Update of access granting for a given area requires to recall all the badges giving access to this area.

Other examples can be identified for similar situations, where the system managing access to the different areas of a company building does not take into account the characteristics of the building layout and of the internal company security policy. Such characteristics can for instance dictate the following rules:

- Staying within a given area for a duration above a pre-determined threshold is a suspicious behavior.
- Transition from a first given area to a second given area without passing through a third given area (typically a security "airlock") is a suspicious behavior.
- Access code recorded on badges must be regularly updated to avoid that stolen or duplicated badges grant access to malicious people

All these types of constraints, such as the constraints illustrated in Figure 1 or the ones illustrated by the former rule list are not properly and efficiently addressed by conventional means. It is the purpose of the present invention to address all these problems.

Within this technical environment, US patent 5,991,411 addresses a different but similar problem, dealing with the adverse use of counterfeit credit cards, access badges, electronic accounts or the like. The solution proposed by this US patent application is based on an history, recorded on a central repository, of the transactions made with the card. As it will be further detailed in the following, this solution clearly departs from the present invention where the access badge dynamically receives, within a given zone of security level Zi, the information required for moving to neighbor security zones.

Objects of the invention

The main object of this invention is to manage the access to protected areas by means of badges and badge readers, where access control is performed both when entering and leaving an area.

It is a further object of this invention to control the time spent by a given badge within a given area.

It is a further object of this invention to dynamically update a secret key used to access an area.

Summary of the invention

The present invention is directed to methods, systems and computer programs as defined in independent claims, for managing access to different areas through badge readers and badges hold by individuals. The present invention is particularly appropriate for environments where different levels of access security are defined.

The method executed in a badge, for having access to different zones with different security levels protected by badge readers, access control being performed both when entering and leaving a protected zone, comprises the steps of:

- receiving from a badge reader, an invitation (*AccessInvite(Ztout))* to have access to a zone, identified by a zone identifier Zout, to which the badge reader gives access;
- determining whether or not the badge is authorized to access the zone Zout identified in the received invitation;
- If the badge is authorized to have access to the identified zone Zout, retrieving a badge identifier IDout associated with the received zone identifier Zout, each badge within a zone Z(i) being identified by an Identifier ID(i);
- issuing to the badge reader, in response to the received invitation, a request *(AccessRequest(ID, IDout, K)*) for having access to the identified zone, said request comprising:

  - a current badge identifier ID;
  - the badge identifier IDout associated with the identified zone Zout; and
  - a current badge key K;
- replacing in the badge, upon reception of an authorization *(AccessGranted(Kout, Tout))* received from the badge reader in response to the access request, to have access to the identified zone Zout during a given period of time Tout:
  - the current badge key K with a received badge key Kout to leave the requested zone Zout;
  - the current badge identifier ID with the badge identifier IDout corresponding to the zone Zout for which an authorization has been received from the badge reader;
  - the current zone identifier Z with the identifier of the zone Zout for which an authorization has been received from the badge reader;
- if the period of time Tout during which the badge holder is authorized to stay in the current zone Zout expires, replacing in the badge:
  - the current badge key Kout by a default badge key Kdef;
  - the current badge identifier IDout by a default badge identifier IDdef;
  - a the current zone identifier Zout by a default zone identifier Zdef.

The method executed in a badge reader, for dynamically managing access to different protected zones with different security levels by means of badges, access control being performed both when entering and leaving a protected zone, comprises the steps of:

- detecting the presence of a badge (*BadgeDetected*);
- issuing to the detected badge, an invitation (*AccessInvite(Zout)*) to have access to a zone, identified by a zone identifier Zout, to which the badge reader gives access;
- receiving from the badge, in response to the invitation, a request *(AccessRequest(ID, IDout, K)* for having access to the zone, comprising:
  - a current badge identifier ID;
  - the badge identifier IDout associated with the zone Zout to which the badge reader gives access; and
  - a current badge key K;

- if the received current badge key K is not equal to a key Kin associated with the zone Zin where the badge reader is located, issuing to the badge, a refusal *(AccessDenied)*;
- if the received key K is equal to the key Kin associated with the zone where the badge reader is located, checking whether or not the badge is authorized to have access the requested zone Zout:
  - If the badge is authorized to have access the requested zone Zout, issuing to the badge an authorization *(AccessGranted(Kout, Tout))* to have access to the requested zone Zout during a given period of time Tout, said authorization comprising:
    - a badge key Kout to leave the zone Zout;
    - a period of time Tout during wich the badge is authorized to stay in the zone Zout.

The method executed in a server connected to one or a plurality of badge readers, for dynamically managing access to different protected zones with different security levels by means of badges and badge readers, access control being performed both when entering and leaving a protected zone, comprises the steps of:

- upon reception from a badge reader of a configuration request (*InitRequest(Zin, Zout)* comprising:
  - a zone identifier Zin, corresponding to the zone where the badge reader is located;
  - a zone identifier Zout, corresponding to the zone to which the badge reader gives access;

  sending back to said badge reader (*InitData(Kin, Kout, IDlist)*):
  - a key Kin associated with the zone Zin where the badge reader is located ;
  - a key Kout associated with the zone Zout to which the badge reader gives access;
  - an IDlist table comprising a list of badge identifiers ID(i) authorized to enter the zone Zout to which the badge reader gives access;

- upon reception from a badge reader, of a message (*Passage(IDto, Zin, Zout)*) informing of an authorization of access and comprising:
  - an identifier IDout of the badge authorized to have access;
  - the zone identifier Zin, corresponding to the zone where the badge reader is located;
  - the zone identifier Zout, corresponding to the zone to which the badge reader gives access;

  updating the Z_ID table by:
  - decrementing the number Pin of badges present in the zone Zin where the badge reader is located; and if the number Pin of badges in the zone where the badge reader is located is equal to zero, sending to the badge reader a new key Kin associated with the zone Zin where the badge reader is located;
  - incrementing the number Pout of badges present in the zone Zout to which the badge reader gives access;

- upon reception from the badge reader, of an intrusion message (*Intrusion(ID, Zin, Zout)*) comprising:
  - a current badge identifier ID,
  - the zone identifier Zin, corresponding to the zone where the badge reader is located;
  - the zone identifier Zout, corresponding to the zone to which the badge reader gives access;

  updating the Z_IDS table by:
  - updating the IDlist table of badges authorized to be in the zone Zin where the badge reader is located by removing the received badge identifier ID from the list, and

    sending the updated IDlist table to the badge reader;
  - decrementing the number Pin of badges present in the zone Zin where the badge reader is located; and if the number Pin of badges in the zone where the badge reader is located is equal to zero, sending to the badge reader a new key Kin associated with the zone Zin where the badge reader is located;

Further embodiments of the invention are provided in the appended dependent claims.

The foregoing, together with other objects, features, and advantages of this invention can be better appreciated with reference to the following specification, claims and drawings.

Brief description of the drawings

The new and inventive features believed characteristics of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further objects and advantages thereof, will best be understood by reference to the following detailed description of an illustrative detailed embodiment when read in conjunction with the accompanying drawings, wherein:

- **Figure 1** represents a building belonging to a private company, with different areas, each of them associated with a specific security level.

- **Figure 2** shows the messages exchanged between badges, badge readers and the central server according to the present invention.

- Figure 3 describes the data used in the messages exchanged between badges, badge readers and the central server according to the present invention.

- **Figure 4** is a flow chart of the method carried out by the badge according to the present invention.

- **Figure 5** is a flow chart of the method carried out by the badge reader according to the present invention.

- **Figure 6** is a flow chart of the method carried out by the central server according to the present invention.

Preferred embodiment of the invention

The following description is presented to enable one or ordinary skill in the art to make and use the invention and is provided in the context of a patent application and its requirements. Various modifications to the preferred embodiment and the generic principles and features described herein will be readily apparent to those skilled in the art. Thus, the present invention is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

Principles of the invention

The method according to the present invention for managing badge access is based on a set of three different types of resources:

- **Badges** : typically owned by employees / visitors. Limited assumptions regarding the hardware implementation of the badge is made in the present invention. It is assumed that a badge comprises :
    - a processor with an associated read/write permanent memory. This memory is loaded with default values (during an initialization phase when leaving the manufacturing),
    - means for managing timers,
    - Input/Output means for controlling exchange of information with a badge reader. Input/Output means are based on any conventional technology, such a magnetic tape, electrical contacts, or wireless communications, and
    - a built-in power source, used to power the whole badge components. Such a power source can typically be implemented with:
        - a conventional battery or
        - photo voltaic cells, or
        - any other conventional means that meet the badge form factor, mechanical and electrical constraints.

      Alternatively, the power source can be external to the badge, the badge being only powered when used, typically
        - from the badge reader through electrical contacts, or

- through radio frequency induction, or
- through any other conventional means that meet the badge form factor, mechanical and electrical constraints.

- **Badge readers** (or readers for short) that grant access to areas. In terms of hardware implementation, it is assumed that the reader includes :
  - a processor with associated memory,
  - means for managing timers,
  - Input/Output means for controlling exchange of information with a badge,
  - a Gate controller for typically opening a door,
  - networking means for controlling exchange of information with a central server, and
  - a power source, typically fed from a conventional power line.

- **Central server :** mainly involved in the distribution of the codes (keys) for delivering access to areas. In terms of hardware implementation, it is assumed that this central server includes :
  - a processor with associated memory,
  - means for managing timers,
  - means for managing a user interface,
  - networking means for controlling exchange of information with a badge reader, and
  - a power source, typically fed a from conventional power line.

The method and system according to the present invention relies on the exchange of information between these three different actors, according to a set of messages as illustrated in Figure 2.

Furthermore the method and system according to the present invention relies on a set of data, within each of the above resources, as described in the Figure 3. Details about this whole set of data are provided in the following sections, whereas a preliminary description is given below:

- Each area or zone protected by the method and system according to the present invention is identified by a unique Zone Identifier Z(i).
- Each zone can be accessed through a Key K(i) hold by a badge and read by a reader.
- Each zone is associated with a maximum time duration T(i) during which a badge is authorised to stay in the zone.
- Each badge within a zone Z(i) is identified by an Identifier ID(i).
- To move from a zone Z(i) to a zone Z(j), a badge with identifier ID(i) must show that it holds the key K(i). If it is the case, the badge receives the key K(j) which allows afterwards to leave the zone Z(j).
- When a zone Z(i) is empty (no badge present in the zone), the server has the possibility to update the key K(i).

From the previous list, the main principles of the present invention are the following:

- A badge reader can't stay indefinitely within a given zone.
- Badge readers are not only used to enter a zone, but also to leave a zone.
- The key used to leave a zone is dynamically passed to the badge when this badge is used to enter in the zone.
- Key are changed when a zone it empty.

All these principles contribute to address the different facets of the security problems previously introduced.

Badge, Reader and Server data

The present invention relies on different methods executed in the badges, the readers and the central servers. These methods use a protocol shared between these objects, based on the primitives described in Figure 2, and on the different pieces of data shown in Figure 3, and specified here below:

Badge Data:

As static data, the badge holds:

- a default key Kdef,
- a default zone identifier Zdef, and
- a default Identifier IDdef.

These pieces of data are used when a badge is first initialized.

As dynamic data, the badge holds:

- a current key K,
- a current zone identifier Z, and
- a current Identifier ID.

These pieces of data correspond to the zone where the badge is currently in.

- A Z_ID table recording pairs of the form (Z(i),ID(i)), each pair informing which zone the badge has access to and under which Identifier this badge is known in this zone.

Badge Reader Data:

As static data, the reader holds:

- a Zone identifier Zin, corresponding to the zone where the badge reader is located,
- a Zone identifier Zout, corresponding to the zone to which the badge reader gives access.

As dynamic data, the reader holds:

- a Key Kin, associated with Zin,
- a Key Kout, associated with Zout,
- An IDlist table recording the list of authorized badge identifier ID(i) for entering the zone Zout.

Server Data:

As dynamic data, the server holds a table Z_IDS, where each record comprises the following fields:

- a zone identifier Z(i),
- the list IDlist(i) of authorised badger identifier for entering in the zone Z(i),
- a population P(i) counting the number of badges present in the zone Z(i),
- a Key K(i), associated with the zone identifier Z(i), and
- a timer T(i) associated with the maximum time a badge can stay in Z(i). If the value of this timer is found equal to 0, then this means that there is no time limitation for staying within the zone Z(i).

The above data are used as arguments of the primitives defined in Figure 2, and exchanged according to the different methods implemented in the badges, in the badge readers, and in the central server:

<u>Methods carried out by the badges, readers, and central server</u>

Method carried out by the badges

The method carried out by the badge is described in Figure 4. This method is implemented as a software program running in the badge processor component, and accessing data in the badge memory component. This method comprises the following steps:

- At step **401**, during an initialization phase, the method starts its operating system.
- At step **402** a self test is executed to check whether or not the badge operates as expected.
- At step **403** a test is performed to check whether or not the self test result is correct.
    - If the self test result is correct, then control is given to step **405**;
    - otherwise control is given to step **404**.
- At step **404**, the badge method aborts if the self test has failed. It is the end of the method. The badge is considered as being inoperative.
- At step **405**, a *StartTimer(BT0)* primitive is issued to the badge timer handler, in order to start a timer BTO. This timer will be used to trigger periodic self tests.
- At step **406** a test is performed to check whether or not the local variable T1 is equal to zero (0).
    - If the local variable T1 is equal to zero (0), then control is given to step **408**;
    - otherwise control is given to step **407**.
- At step **407**, a *StartTimer(BT1)* primitive is issued to the badge timer handler, in order to start a timer BT1, with a time-out duration equal to T1. This timer will be used to trigger key validity: the key will be reset if this timer reaches a time-out condition (see step **410**).
- At step **408**, the badge method is in its default state, waiting for events corresponding to the reception of primitives (see steps **409**, **410**, **411**, and **414**).
- At step **409**, a *TimeOut(BTO)* primitive is received from the badge timer handler. Control is then given to step **402** for running a periodic self test.

- At step **410**, a *TimeOut(BT1)* primitive is received from the badge timer handler. Control is given to step **429** for resetting the current key.
- At step **411**, an *AccessUpdate(Z_ID, K, Z, ID)* primitive is received from the badge reader.
- At step **412**, the badge configuration data are updated as follows:
  - by replacing the current Z_ID table with the first argument of the received *AccessUpdate(Z_ID, K, Z, ID)* primitive,
  - by replacing the badge current key K by the second argument of the received *AccessUpdate(Z_ID, K, Z, ID)* primitive,
  - by replacing the badge current zone identifier Z by the third argument of the received *AccessUpdate(Z_ID, K, Z, ID)* primitive, and
  - by replacing the badge current identifier ID by the fourth argument of the received *AccessUpdate(Z_ID, K, Z, ID)* primitive.
- At step **413**, a *StopTimer(BTO)* primitive and a *StopTimer(BT1)* primitive are issued to the badge timer handler, in order to stop the timers BTO and BT1. Then control is given back to the step **429**.
- At step **414**, an *AccessInvite(Zto)* primitive is received from the badge reader.
- At step **415**, a test is performed to check whether or not the zone identifier Zto is found present in the Z_ID table.
  - If the zone identifier Zto is found present in the Z_ID table, then control is given to step **416**;
  - otherwise control is given to step **417**.
- At step **416**, the identifier IDto associated with the zone identifier Zto is retrieved from the Z_ID table. Then control is given to step **418**.
- At step **417**, the identifier IDto is initialized with a null value (0).
- At step **418**, an *AccessRequest(ID, IDto, K)* primitive is issued to the badge reader.
- At step **419**, a *StartTimer(BT2)* primitive is issued to the badge timer handler, in order to start a timer BT2. This timer will be used to trigger the absence of badge reader feedback.
- At step **420**, the badge method is in a transient state, waiting for a feedback from the badge reader (see steps **421**, **422**, **423**, and **426**).
- At step **421**, a *TimeOut(BT2)* primitive is received from the badge timer handler. Control is then given to step **402** for running a periodic self test.

- At step **422**, an *InvalidAccess* primitive is received from the badge reader. Then control is given to step **425**.
- At step **423**, an *AccessGranted(Kout, Tout)* primitive is received from the badge reader.
- At step **424**,
  - the current key K takes the value of the received key Kout,
  - the current identifier ID takes the value of the identifier IDto,
  - the current zone identifier Z takes the value of the zone identifier Zto, and
  - finally a local variable T1 is set equal to the received value Tout.
- At step **425**, a *StopTimer(BT2)* primitive is issued to the badge timer handler, in order to stop the timer BT2. Then control is given back to the step **402**.
- At step **426**, an *AccessDenied* primitive is received from the badge reader.
- At step **427**, all the badge configuration data are reset.
- At step **428**, a *StopTimer(BT0)* primitive, a *StopTimer(BT1)* primitive, and a *StopTimer(BT2)* primitive are issued to the badge timer handler, in order to stop the timers BT0, BT1, and BT2.
- At step **429**, default values are assigned to the variables associated with the badge (as it is done when a brand new badge leaves manufacturing):
  - the current key K takes the value of the default key Kdef,
  - the current identifier ID takes the value of the default identifier Iddef,
  - the current zone identifier Z takes the value of the default zone identifier Zdef, and
  - finally a local variable T1 is set equal to the zero value (0).

  Then control is given back to the initial step **401**.

Method carried out by the badge readers

The method carried out by the badge reader is described in Figure 5. This method is implemented as a software program running in the badge reader processor component, and accessing data in the badge reader memory component. This method comprises the following steps:

- At step **501**, during an initialization phase, the badge reader method starts its operating system and loads the zone identifiers Zin and Zout from its static configuration data.
- At step **502**, a self test is executed to check that the badge reader operates as expected.
- At step **503,** a test is performed to check whether or not the self test result is correct.
  - If the self test result is correct, then control is given to step **505**;
  - otherwise control is given to step **504**.
- At step **504**, the badge reader methods aborts if the self test has failed. It is the end of the method; The badge reader is considered as being inoperative.
- At step **505**, an *InitRequest(Zin, Zout)* primitive is issued to the server, in order to receive initial configuration data.
- At step **506**, a *StartTimer(RT0)* primitive is issued to the badge reader timer handler, in order to start a timer RT0. This timer will be used to trigger the absence of server feedback.
- At step **507**, the badge reader method is in a transient state, waiting for the server feedback (see steps **508**, and **509**).
- At step **508**, a *TimeOut(RT0)* primitive is received from the badge reader timer handler. Control is then given to step **502** for running a periodic self test.
- At step **509**, an *InitData(Kin, Kout, Idlist)* primitive is received from the server.
- At step **510**, a *StopTimer(RT0)* primitive and a *StartTimer(RT1)* primitive are issued to the badge reader timer handler, in order to stop the timer RT0, and to start the timer RT1 covering the absence of server refresh.
- At step **511**, the badge reader configuration data Kin, Kout and IDlist are initialized with the parameters of the primitive *InitData(Kin, Kout, Idlist)* received at step **509**.
- At step **512**, the badge reader method is in its default state, waiting for events corresponding to the reception of primitives (see steps **513**, **514**, **516**, and **518**).
- At step **513**, a *TimeOut(RT1)* primitive is received from the badge reader timer handler. Control is then given to step **502** for running a periodic self test.
- At step **514**, an *InitData(Kin, Kout, Idlist)* primitive is received from the server.
- At step **515**, a *StartTimer(RT1)* primitive is issued to the badge reader timer handler, in order to restart the timer RT1 covering the absence of server refresh. Then control is given to step **511**.

- At step **516**, an *UpdateBadge(Z_ID, K, Z, ID)* primitive is received from the server.
- At step **517**, an *AccessUpdate(Z_ID, K, Z, ID)* primitive is issued to the badge. Then control is given to step **512**.
- At step **518**, a *BadgeDetected* primitive is received from the badge reader I/O Controller, as a notification that a badge has been detected.
- At step **519**, an *AccessInvite(Zto)* primitive is issued to the badge.
- At step **520**, a *Freeze(RT1)* primitive and a *StartTimer(RT2)* primitive are issued to the badge reader timer handler, in order to freeze the timer RT1, and to start the timer RT2 covering the absence of badge feedback.
- At step **521**, the badge reader method is in a transient state, waiting for the badge reader feedback (see steps **522**, and **524**).
- At step **522**, a *TimeOut(RT2)* primitive is received from the badge reader timer handler.
- At step **523**, an *Unfreeze(RT1)* primitive is issued to the badge reader timer handler, in order to unfreeze the timer RT1. Then control is given to step **512**.
- At step **524**, an *AccessRequest(ID, IDto, K)* primitive is received from the badge.
- At step **525**, a *StopTimer(RT2)* primitive is issued to the badge reader timer handler, in order to stop the timer RT2.
- At step **526**, a test is performed to check whether or not the key K received as last parameter of the *AccessRequest(ID, IDto, K)* primitive received at step **524** is equal to the local key Kin.
  - If the key K received as last parameter of the *AccessRequest(ID, IDto, K)* primitive received at step **524** is equal to the local key Kin, then control is given to step **529**;
  - otherwise control is given to step **527**.
- At step **527**, an *AccessDenied* primitive is issued to the badge.
- At step **528**, an *Intrusion(ID, Zin, Zout)* primitive is issued to the server. Then control is given to step **501**.
- At step **529**, a test is performed to check whether or not the identifier IDto is found within the IDlist table.
  - If the identifier IDto is found within the IDlist table, then control is given to step **532**;
  - otherwise control is given to step **530**.

- At step **530**, an *InvalidAccess* primitive is issued to the badge.
- At step **531**, the badge holder is warned through conventional means, such as, but not limited to, an audible message, or a visible message. Then control is given to step **523**.
- At step **532**, an *AccessGranted(Kout, Tout)* primitive is issued to the badge.
- At step **533**, a *Passage(IDto, Zin, Zout)* primitive is issued to the server.
- At step **534**, an *OpenGate* primitive is issued to the gate controller, for giving access to the badge holder. Then control is given to step **523**.

Method carried out by the central server

The method carried out by the central server is described in Figure 6. This method is implemented as a software program running in the server processor component, and accessing data in the server memory component. This method comprises the following steps :

- At step **601**, during an initialization phase, the server method starts its operating system.
- At step **602**, a self test is executed to check that the server operates as expected.
- At step **603**, a test is performed to check if the self test result is correct.
  - If the self test result is correct, then control is given to step **605**;
  - otherwise control is given to step **604**.
- At step **604**, the server method aborts as the self test has failed. It is the end of the method, the server is considered as being no longer operative.
- At step **605**, the configuration data is initialized by loading in memory the Z_IDS table.
- At step **606**, an *InitData(Kin, Kout, IDlist)* primitive is issued to the badge reader.
- At step **607**, a *StartTimer(ST0)* primitive is issued to the server timer handler, in order to start a timer STO. This timer will be used to trigger periodic self tests.
- At step **608**, the server method is in its default state, waiting for events corresponding to the reception of primitives (see steps **609**, **610**, **612**, **615**, and **617**).
- At step **609**, a *TimeOut(STO)* primitive is received from the server timer handler. Control is then given to step **602** for running a periodic self test.
- At step **610**, an InitRequest(Zin, Zout) primitive is received from the badge reader.

- At step **611**, an *InitData(Kin, Kout, IDlist)* primitive is issued to the badge reader.
  - The parameter Kin is retrieved from the Z_IDS table as the Key field of the record containing a zone identifier equal to Zin.
  - The parameter Kout is retrieved from the Z_IDS table as the Key field of the record containing a zone identifier equal to Zout.
  - The IDlist parameter is retrieved from the Z_IDS table as the IDlist field of the record containing a zone identifier equal to Zout.
- At step **612**, a *Passage(IDto, Zin, Zout)* primitive is received from the badge reader.
- At step **613**, the Z_IDS table is updated:
  - by decrementing the Pin field in the record where the zone identifier is equal to Zin, and
  - by incrementing the Pout field in the record where the zone identifier is equal to Zout.
- At step **614**, a test is performed to check whether or not the Pin variable is equal to zero (0).
  - If the Pin variable is equal to zero (0), then control is given to step **620**;
  - otherwise control is given to step **608**.
- At step **615**, an *Intrusion(ID, Zin, Zout)* primitive is received from the badge reader.
- At step **616**, the Z_IDS table is updated
  - by removing ID in the Idlist field, and
  - by decrementing the Pin field in the record where the zone identifier is equal to Zin.
  -

  Then control is given to step **614**.
- At step **617**, an *UserUpdate(Z_ID, K, Z, ID)* primitive is received from the user interface controller in the server.
- At step **618**, the Z_IDS table is updated for reflecting the update of user access rights, as specified in the received primitive *UserUpdate(Z_ID, K, Z, ID)*: for each record (Z*, ID*) of the Z_ID table, the specified identifier ID* is added to the IDlist field within the Z_IDS record whose the zone identifier is equal to Z*.
- At step **619**, an *UpdateBadge(Z_ID, K, Z, ID)* primitive is issued to the badge reader. Then control is given to step **608**.

- At step **620**, a new key Kin is generated. This new key can be based on any conventional means used for generating random numbers.
- At step **621**, an *InitData(Kin, Kout, Idlist)* primitive is issued to the badge reader. Then control is given to step **608**.

Initialization step

It is worth noticing that these methods include an initialization step allowing to first define the table Z_ID in the badge and the table Z_IDS in the server. This initialization step is conducted through a dedicated reader, such as the reader shown in Figure 1 at the boundary between the lobby Z0 and the security center Z3.

Primitives

The different primitives used in the present invention are summarized in the following table, where the words “badge”, “reader” and “server” have been respectively shortened into “B”, “R” and “S”:

Primitive	From	To	Purpose / comment
StartTimer (xT)	Processor in B/R/S	Timer in B/R/S	For starting a timer whose time-out is xT
StopTimer	Processor in B/R/S	Timer in B/R/S	For stopping the started timer with time-out xT
TimeOut(xT)	Processor in B/R	Timer in B/R	For notifying that the time-out duration has been elapsed
Freeze(RT)	Processor in R	Timer in R	For freezing the started timer with time-out RT
Unfreeze(RT)	Processor in R	Timer in R	For restarting the freezed timer with time-out RT
BadgeDetected	I/O Ctrl in R	Processor in R	For notifying that a badge is detected in the reader
OpenGate	Processor in R	Gate Ctlr in R	For asking to open the gate
AccessInvite(Zto)	Processor in R	Processor in B	For inviting the badge to ask for access to zone Zto. This message is relayed through the I/O Ctrl of both R and B
AccessRequest(ID, IDto,K)	Processor in B	Processor in R	For requesting access to a zone. ID is the current badge identifier (in Zin), IDto is the badge ID in the target zone, and K is the key of the target zone.

AccessDenied	Processor in R	Processor in B	For denying zone access, due to a wrong parameter K in the access request
InvalidAccess	Processor in R	Processor in B	For invalidating zone access, due to a wrong IDto parameter in the access request
AccessGranted(Kout, Tout)	Processor in R	Processor in B	For giving zone access to Zout, associated with Key Kout and timer Tout.
AccessUpdate(Z_ID,K,Z,ID)	Processor in R	Processor in B	For updating data in the Z_ID table.
InitRequest(Zin,Zout)	Processor in R	Processor in S	For requesting initialization data for the reader from Zin to Zout.
InitData(Kin, Kout, IDlist)	Processor in S	Processor in R	For passing initialization data to the reader from Zin to Zout.
Intrusion(ID, Zin, Zout)	Processor in R	Processor in S	For notifying an intrusion of badge ID (same case as for AccessDenied)
Passage(IDto, Zin, Zout)	Processor in R	Processor in S	For notifying a passage from Zin to Zout of the badge IDto.
UpdateBadge(Z_ID,K,Z,ID)	Processor in S	Processor in R	For updating data in the Z_ID table.
UserUpdate(Z_ID, K, Z, ID)	User I/F in S	Processor in S	For updating data in the Z_ID table.

For the above primitives, their parameters can be advantageously encrypted through conventional ciphering means. No assumption is taken regarding both the technology that can be used for information ciphering and the management of the associated secret keys, as they are both considered to stay beyond the scope of the present invention.

Alternate Embodiment

In an alternate embodiment of the present invention, the key K associated to a given zone can furthermore be instantiated by badge. This can be achieved, when a key K is exchanged between a badge reader and a badge with identifier ID, by replacing the key K by the result of a hashing function fed with both the zone key K and the badge identifier ID: Hash(K,ID). This new key K'=Hash(K,ID) will be unique for each pair (K,ID) and can replace the key parameter K in the primitives AccessRequest(ID, IDto, K), AccessGranted(Kout, Tout), AccessUpdate(Z_ID,K,Z,ID). Without requiring additional memory field in the different tables and data associated to the badges and badge readers, this new key K' allows to keep the zone key K hidden. Outputs of hashing functions have a fixed-length, typically 128 bits for MD5 (See: "The MD5

Message-Digest Algorithm" RFC 1321 from R.Rivest), or 160 bits for SHA-1 (See "Secure Hash Algorithm 1" RFC 3174).

While the invention has been particularly shown and described with reference to a preferred embodiment, it will be understood that various changes in form and detail may be made therein without departing from the spirit, and scope of the invention.

Claims

What is claimed is:

1. A method executed in a badge, for having access to different zones with different security levels protected by badge readers, access control being performed both when entering and leaving a protected zone, said method comprising the steps of:

- receiving from a badge reader, an invitation (*AccessInvite(Ztout))* to have access to a zone, identified by a zone identifier Zout, to which the badge reader gives access;
- determining whether or not the badge is authorized to access the zone Zout identified in the received invitation;
- If the badge is authorized to have access to the identified zone Zout, retrieving a badge identifier IDout associated with the received zone identifier Zout, each badge within a zone Z(i) being identified by an Identifier ID(i);
- issuing to the badge reader, in response to the received invitation, a request *(AccessRequest(ID, IDout, K)*) for having access to the identified zone, said request comprising:
  - a current badge identifier ID;
  - the badge identifier IDout associated with the identified zone Zout; and
  - a current badge key K;
- replacing in the badge, upon reception of an authorization *(AccessGranted(Kout, Tout))* received from the badge reader in response to the access request, to have access to the identified zone Zout during a given period of time Tout:
  - the current badge key K with a received badge key Kout to leave the requested zone Zout;
  - the current badge identifier ID with the badge identifier IDout corresponding to the zone Zout for which an authorization has been received from the badge reader;
  - the current zone identifier Z with the identifier of the zone Zout for which an authorization has been received from the badge reader;
- if the period of time Tout during which the badge holder is authorized to stay in the current zone Zout expires, replacing in the badge:

- the current badge key Kout by a default badge key Kdef;
- the current badge identifier IDout by a default badge identifier IDdef;
- a the current zone identifier Zout by a default zone identifier Zdef.

2. The method according to the preceding claim wherein the step of determining whether or not the badge is authorized to access the zone Zout identified in the received invitation, comprises the further step of :

- checking whether or not the received zone identifier Zout is listed in a Z_ID table stored in the badge, said a Z_ID table comprising a list of one or a plurality of zone identifiers Z(i), each zone identifier Z(i) being associated with a badge identifier ID(i), in order to determine which zones the badge is authorized to have access to and under which identifier ID(i) this badge is known in each zone Z(i).

3. The method according to any one of the preceding claims wherein the step of determining whether or not the badge is authorized to access the zone Zout identified in the received invitation, comprises the further step of:

- If the badge is not authorized to have access to the identified zone Zout, assigning a predetermined value to the badge identifier in order to prevent the badge holder to have access to the identified zone Zout;

4. The method according to any one of the preceding claims comprising the further step of:

- receiving from the badge reader in response to the access request, a refusal *(AccessDenied*) to have access to the requested zone Zout due to a current badge key K different from the badge key Kin associated with the zone Zin where the badge reader is located;
- replacing in the badge:
  - the current badge key K by a default badge key Kdef;
  - the current badge identifier ID by a default badge identifier IDdef;
  - the current zone identifier Z by a default zone identifier Zdef.

5. The method according to any one of the preceding claims comprising the further step of:

- receiving from the badge reader, in response to an access request, a refusal (*InvalidAccess*) to have access to the requested zone Zout due to an unknown badge identifier ID or due to a not authorized badge identifier for the requested zone Zout.

6. The method according to any one of the preceding claims comprising the further step of:

- receiving from the badge reader an access update (*AccessUpdate(Z_ID, K, Z, ID)* primitive) for replacing in the badge:
    - the current table Z_ID table with a new table;
    - the current badge key K by a new badge key;
    - the current zone identifier Z by a new zone identifier; and
    - the current badge identifier ID by a new badge identifier.

7. The method according to any one of the preceding claims wherein the step of replacing in the badge, upon reception of an authorization *(AccessGranted(Kout, Tout))* received from the badge reader in response to the access request, to have access to the identified zone Zout during a given period of time Tout, comprises the further step of:

- starting a timer with said given period of time Tout as time-out.

8. A badge comprising means adapted for carrying out the method according to any one of the preceding claims.

9. A computer program comprising instructions for carrying out the method according to any one of claims 1 to 7 when said computer program is executed in a badge.

10. A method executed in a badge reader, for dynamically managing access to different protected zones with different security levels by means of badges, access control being performed both when entering and leaving a protected zone, said method comprising the steps of:

- detecting the presence of a badge (*BadgeDetected*);
- issuing to the detected badge, an invitation (*AccessInvite(Zout)*) to have access to a zone, identified by a zone identifier Zout, to which the badge reader gives access;
- receiving from the badge, in response to the invitation, a request *(AccessRequest(ID, IDout , K)* for having access to the zone, comprising:
  - a current badge identifier ID;
  - the badge identifier IDout associated with the zone Zout to which the badge reader gives access; and
  - a current badge key K;
- if the received current badge key K is not equal to a key Kin associated with the zone Zin where the badge reader is located, issuing to the badge, a refusal *(AccessDenied*);
- if the received key K is equal to the key Kin associated with the zone where the badge reader is located:
  - checking whether or not the badge is authorized to have access the requested zone Zout;
  - If the badge is authorized to have access the requested zone Zout, issuing to the badge an authorization *(AccessGranted(Kout, Tout)*) to have access to the requested zone Zout during a given period of time Tout, said authorization comprising:
    - a badge key Kout to leave the zone Zout;
    - a period of time Tout during wich the badge is authorized to stay in the zone Zout.

11. The method according to the preceding claim comprising the preliminary step of:

- storing in memory:

   - a zone identifier Zin, corresponding to the zone where the badge reader is located;
   - a zone identifier Zout, corresponding to the zone to which the badge reader gives access;
- sending a configuration request (*InitRequest(Zin, Zout)*) to a server comprising:
   - said zone identifier Zin, corresponding to the zone where the badge reader is located;
   - said zone identifier Zout, corresponding to the zone to which the badge reader gives access;
- receiving from the server (*InitData(Kin, Kout, IDlist)*) in response to the configuration request and storing:
   - a key Kin, associated with the identifier Zin of the zone where the badge reader is located;
   - a key Kout, associated with the identifier Zout of the zone to which the badge reader gives access;
   - an IDlist table comprising a list of badge identifiers ID(i) authorized to enter the zone Zout to which the badge reader gives access;

12. The method according to any one of claims 10 to 11 wherein the step of checking whether or not the badge is authorized to have access to the requested zone Zout, comprises the further step of:

- checking whether or not the badge identifier IDout is recognized in a IDlist table comprising a list of authorized badges for the requested zone Zout.

13. The method according to any one of claims 10 to 12 wherein the step of issuing to the badge, a refusal *(AccessDenied* primitive) to have access to the requested zone Zout, comprises a further step of:

- sending to a server an intrusion message (*Intrusion(ID, Zin, Zout))* comprising:
   - the received current badge identifier ID;
   - the zone identifier Zin, corresponding to the zone where the badge reader is located;

- the identifier Zout corresponding to the zone to which the badge reader gives access.

14. The method according to any one of claims 10 to 13 comprising the further step of:

- receiving from the server a request (*UpdateBadge(Z_ID, K, Z, ID)*) for updating the badge, comprising:
  - a Z_ID table;
  - a badge key K;
  - a zone identifier Z;
  - a badge identifier ID;
- issuing to the badge an access update (*AccessUpdate(Z_ID, K, Z, ID)*) for replacing in the badge:
  - the current Z_ID table by the received Z_ID table;
  - the current badge key K by the received badge key K;
  - the current zone identifier Z by the received zone identifier Z;
  - the current badge identifier by the received badge identifier ID.

15. The method according to any one of claims 10 to 14 wherein the step of issuing to the badge an authorization to have access access to the zone Zout during a given period of time Tout *(AccessGranted(Kout, Tout)*), comprises the further step of:

- sending to the server a message (a *Passage(IDout, Zin, Zout)*) comprisng:
  - the identifier IDout of the badge authorized to have access;
  - the zone identifier Zin, corresponding to the zone where the badge reader is located;
  - the zone identifier Zout, corresponding to the zone to which the badge reader gives access;
- giving access (*OpenGate*) to the badge holder.

16. The method according to any one of claims 10 to 15 wherein the step of checking whether or not the badge is authorized to have access to the requested zone Zout, comprises the further step of:

If the badge is not authorized to have access to the requested zone Zout:

- issuing to the badge a refusal (*InvalidAccess*) to have access to the requested zone Zout.

17. The method according to any one of claims 10 to 16 wherein the badge key K received from or sent to the badge is unique to each badge.

18. The method according to any one of claims 10 to 16 wherein the badge key K received from or sent to the badge is the result of a hashing function Hash(K,ID) fed with both a key specific to the zone and the identifier ID of the badge.

19. A badge reader comprising means adapted for carrying out the method according to any one of claims 10 to 18.

20. A computer program comprising instructions for carrying out the method according to any one of claims 10 to 18 when said computer program is executed by a badge reader.

21. A method executed in a server connected to one or a plurality of badge readers, for dynamically managing access to different protected zones with different security levels by means of badges and badge readers, access control being performed both when entering and leaving a protected zone, said method comprising the steps of:

- upon reception from a badge reader of a configuration request (*InitRequest(Zin, Zout)* comprising:
  - a zone identifier Zin, corresponding to the zone where the badge reader is located;
  - a zone identifier Zout, corresponding to the zone to which the badge reader gives access;

  sending back to said badge reader (*InitData(Kin, Kout, IDlist)*):
  - a key Kin associated with the zone Zin where the badge reader is located ;

- a key Kout associated with the zone Zout to which the badge reader gives access;
- an IDlist table comprising a list of badge identifiers ID(i) authorized to enter the zone Zout to which the badge reader gives access;

- upon reception from a badge reader, of a message (*Passage(IDto, Zin, Zout)*) informing of an authorization of access and comprising:
  - an identifier IDout of the badge authorized to have access;
  - the zone identifier Zin, corresponding to the zone where the badge reader is located;
  - the zone identifier Zout, corresponding to the zone to which the badge reader gives access;

  updating the Z_ID table by:
  - decrementing the number Pin of badges present in the zone Zin where the badge reader is located; and if the number Pin of badges in the zone where the badge reader is located is equal to zero, sending to the badge reader a new key Kin associated with the zone Zin where the badge reader is located;
  - incrementing the number Pout of badges present in the zone Zout to which the badge reader gives access;

- upon reception from the badge reader, of an intrusion message (*Intrusion(ID, Zin, Zout)*) comprising:
  - a current badge identifier ID,
  - the zone identifier Zin, corresponding to the zone where the badge reader is located;
  - the zone identifier Zout, corresponding to the zone to which the badge reader gives access;

  updating the Z_IDS table by:
  - updating the IDlist table of badges authorized to be in the zone Zin where the badge reader is located by removing the received badge identifier ID from the list, and

    sending the updated IDlist table to the badge reader;

  - decrementing the number Pin of badges present in the zone Zin where the badge reader is located; and if the number Pin of badges in the zone where the badge reader is located is equal to zero, sending to the badge reader a new key Kin associated with the zone Zin where the badge reader is located;

22. The method according to claim 21 comprising the preliminary step of:

- loading in memory a Z_IDS table wherein each zone is associated with:
  - a zone identifier Z(i);
  - a list IDlist(i) of badge identifiers authorized to enter in the zone Z(i);
  - a number P(i) of badges present in the zone Z(i);
  - a key K(i), associated with the zone identifier Z(i); and
  - a period of time corresponding to the maximum time a badge is authorized to stay in the zone Z(i).

23. The method according to any one of claims 21 to 22 comprising the further steps of of:

upon reception of a user update (*UserUpdate(Z_ID, K, Z, ID)* from a user interface, comprising:
  - an updated Z_ID table;
  - an updated badge key K;
  - an updated zone identifier Z;
  - an updated badge identifier ID.
- updating the Z_IDS table accordingly.

24. The method according to any one of claims 21 to 23 comprising the further step of of:

- sending to the badge reader in charge of badge initialisation a request for updating the badge (*UpdateBadge(Z_ID, K, Z, ID)*), comprising:
  - the updated Z_ID table;
  - a badge key K;

- a zone identifier Z;
- a badge identifier ID.

25. A server comprising means adapted for carrying out the method according to any one of claims 21 to 24.

26. A computer program comprising instructions for carrying out the method according to any one of claims 21 to 24 when said computer program is executed on a server.

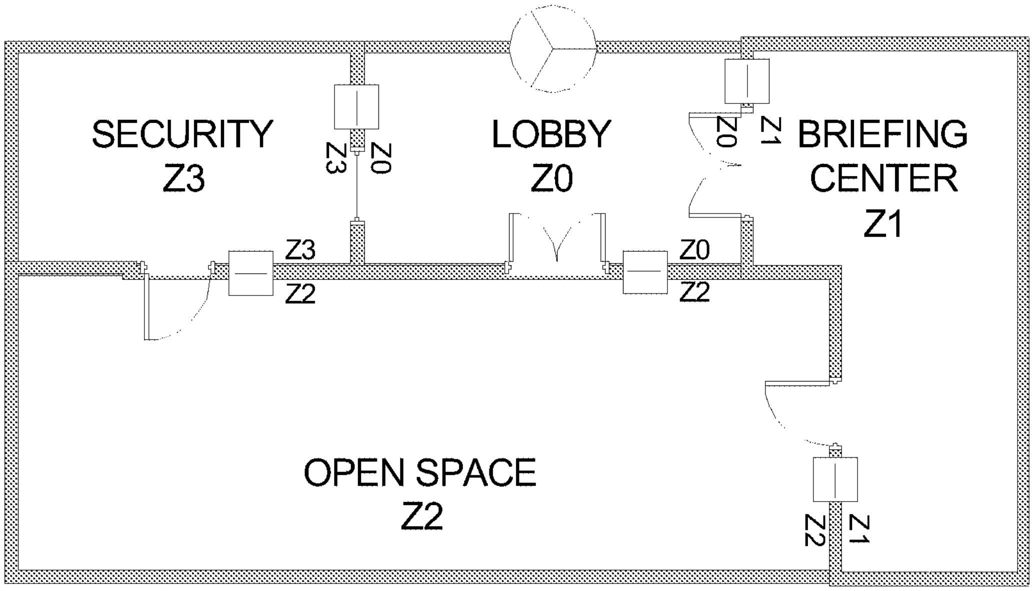

SECURITY
Z3
LOBBY
Z0
BRIEFING
CENTER
Z1
OPEN SPACE
Z2
Z3
Z0
Z3
Z2
Z0
Z2
Z0
Z1
Z2
Z1

FIG 1

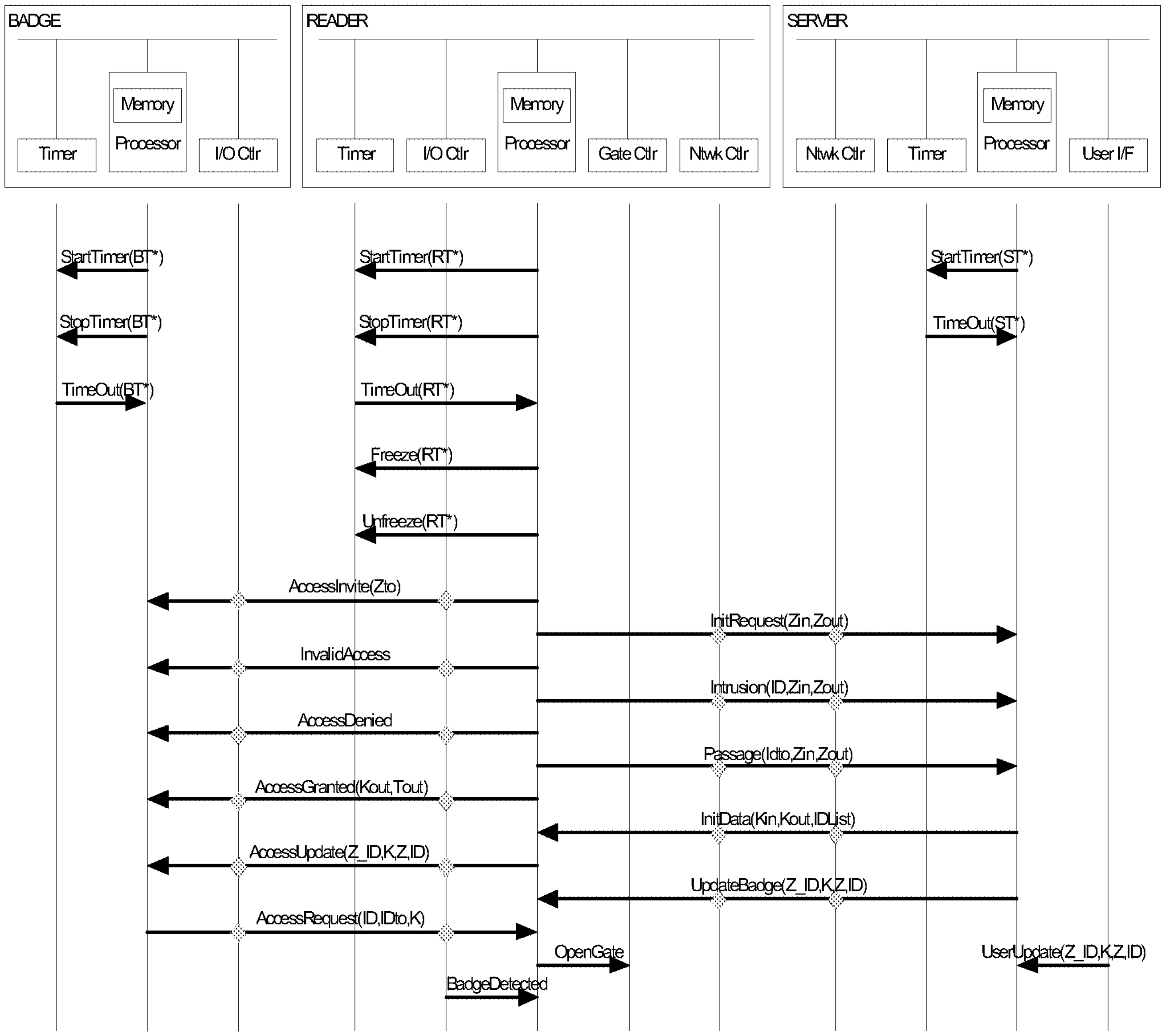

BADGE
Timer
Memory
Processor
I/O Ctlr
READER
Timer
I/O Ctlr
Memory
Processor
Gate Ctlr
Ntwk Ctlr
SERVER
Ntwk Ctlr
Timer
Memory
Processor
User I/F
StartTimer(BT*)
StopTimer(BT*)
TimeOut(BT*)
StartTimer(RT*)
StopTimer(RT*)
TimeOut(RT*)
Freeze(RT*)
Unfreeze(RT*)
StartTimer(ST*)
TimeOut(ST*)
AccessInvite(Zto)
InitRequest(Zin,Zout)
InvalidAccess
Intrusion(ID,Zin,Zout)
AccessDenied
Passage(Idto,Zin,Zout)
AccessGranted(Kout,Tout)
InitData(Kin,Kout,IDList)
AccessUpdate(Z_ID,K,Z,ID)
UpdateBadge(Z_ID,K,Z,ID)
AccessRequest(ID,IDto,K)
OpenGate
UserUpdate(Z_ID,K,Z,ID)
BadgeDetected

FIG2

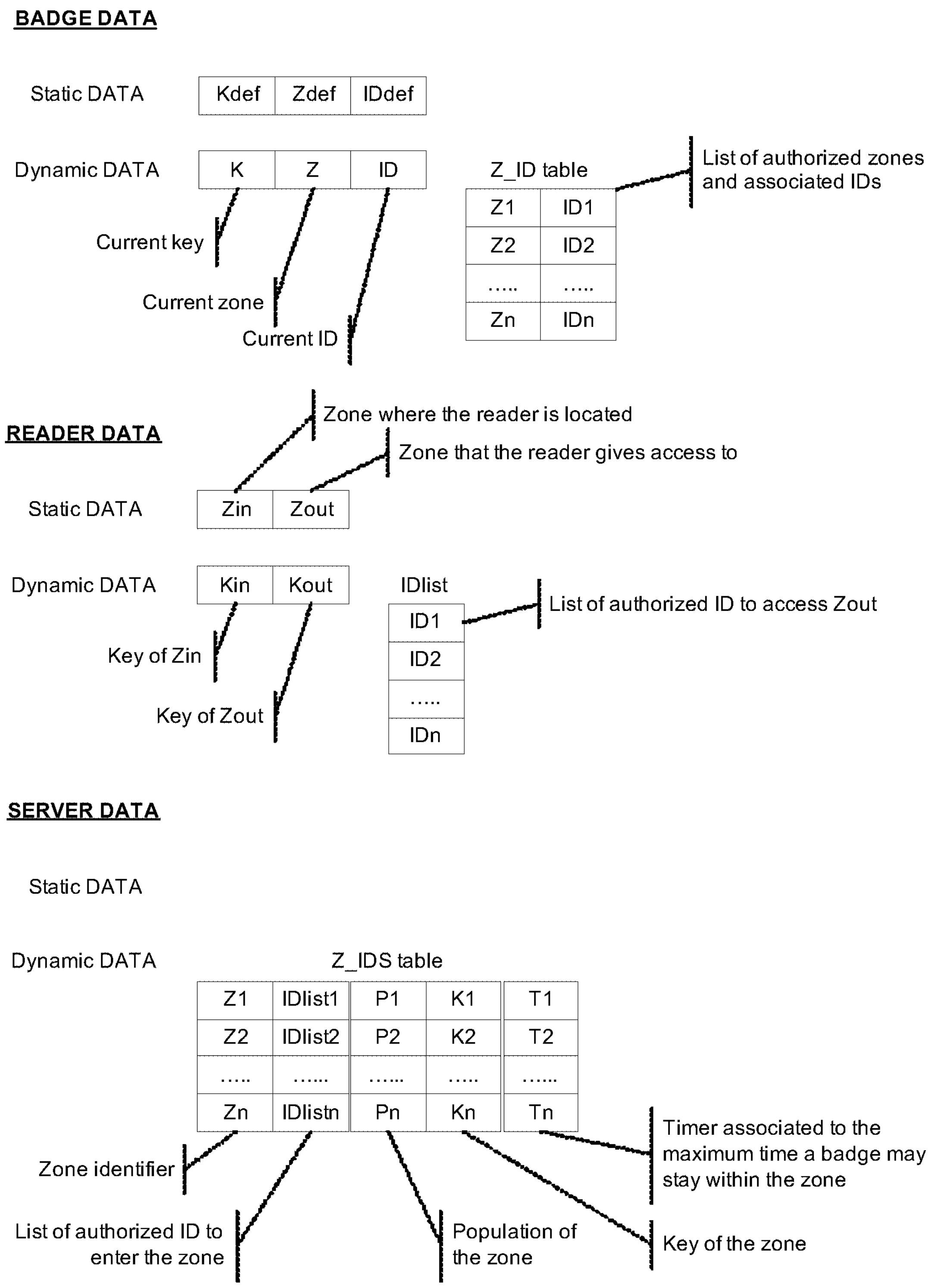
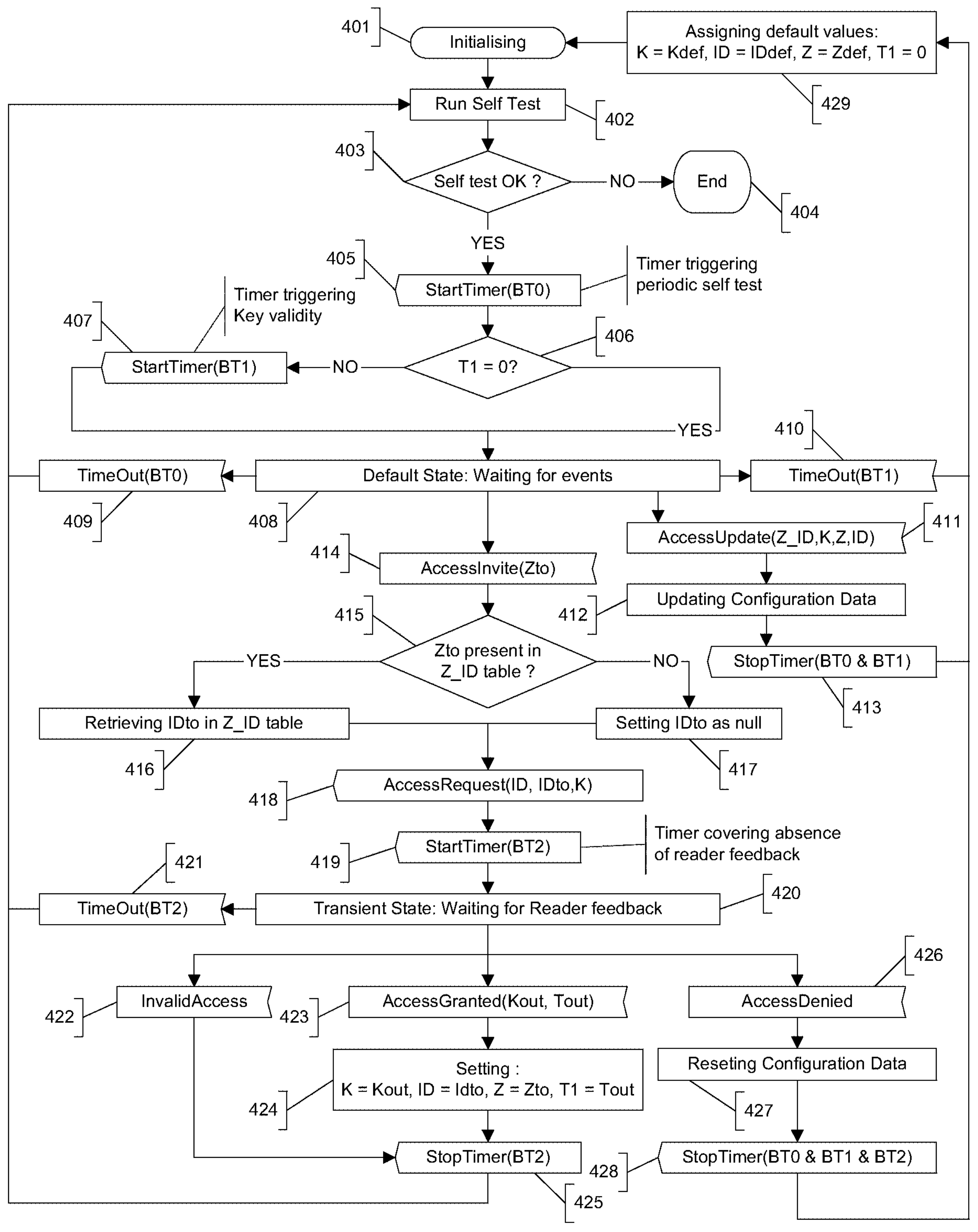

BADGE DATA
Static DATA
Kdef
Zdef
IDdef
Dynamic DATA
K
Z
ID
Current key
Current zone
Current ID
Z_ID table
Z1 ID1
Z2 ID2
….. …..
Zn IDn
List of authorized zones and associated IDs
READER DATA
Zone where the reader is located
Zone that the reader gives access to
Static DATA
Zin
Zout
Dynamic DATA
Kin
Kout
Key of Zin
Key of Zout
IDlist
ID1
ID2
…..
IDn
List of authorized ID to access Zout
SERVER DATA
Static DATA
Dynamic DATA
Z_IDS table
Z1 IDlist1 P1 K1 T1
Z2 IDlist2 P2 K2 T2
….. …... …... ….. …...
Zn IDlistn Pn Kn Tn
Zone identifier
List of authorized ID to enter the zone
Population of the zone
Key of the zone
Timer associated to the maximum time a badge may stay within the zone

FIG 3

BADGE METHOD



401
Initialising
Assigning default values:
K = Kdef, ID = IDdef, Z = Zdef, T1 = 0
429
Run Self Test
402
403
Self test OK ?
NO
End
404
YES
405
StartTimer(BT0)
Timer triggering periodic self test
Timer triggering Key validity
407
StartTimer(BT1)
NO
T1 = 0?
406
YES
410
TimeOut(BT0)
Default State: Waiting for events
TimeOut(BT1)
409
408
AccessUpdate(Z_ID,K,Z,ID)
411
414
AccessInvite(Zto)
415
412
Updating Configuration Data
Zto present in Z_ID table ?
YES
NO
StopTimer(BT0 & BT1)
413
Retrieving IDto in Z_ID table
Setting IDto as null
416
417
AccessRequest(ID, IDto,K)
418
StartTimer(BT2)
419
Timer covering absence of reader feedback
421
TimeOut(BT2)
Transient State: Waiting for Reader feedback
420
426
InvalidAccess
AccessGranted(Kout, Tout)
AccessDenied
422
423
Setting :
K = Kout, ID = Idto, Z = Zto, T1 = Tout
Reseting Configuration Data
424
427
StopTimer(BT2)
428
StopTimer(BT0 & BT1 & BT2)
425

FIG 4

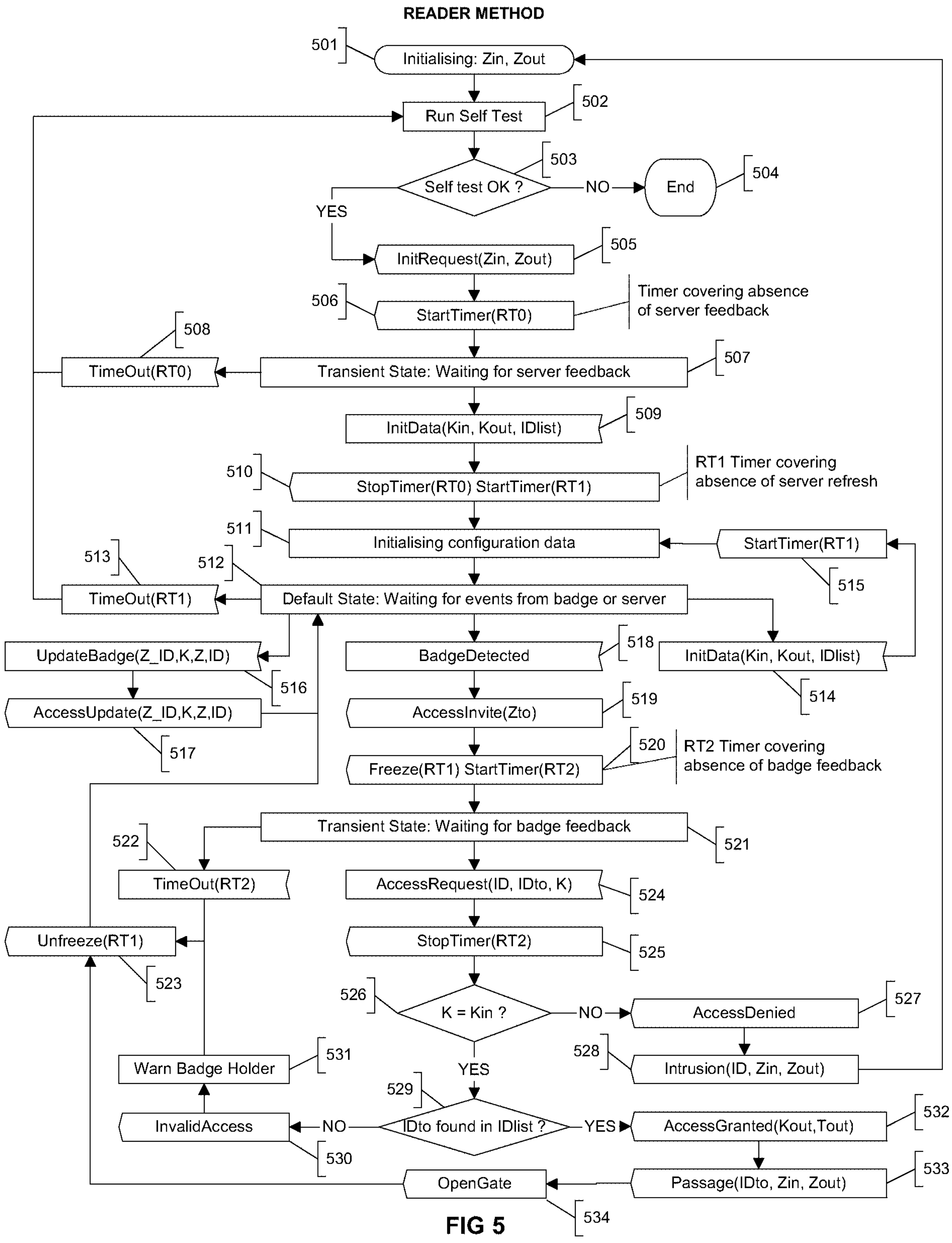

READER METHOD
501
Initialising: Zin, Zout
Run Self Test
502
503
Self test OK ?
NO
End
504
YES
InitRequest(Zin, Zout)
505
506
StartTimer(RT0)
Timer covering absence of server feedback
508
TimeOut(RT0)
Transient State: Waiting for server feedback
507
InitData(Kin, Kout, IDlist)
509
510
StopTimer(RT0) StartTimer(RT1)
RT1 Timer covering absence of server refresh
511
Initialising configuration data
StartTimer(RT1)
515
513
512
TimeOut(RT1)
Default State: Waiting for events from badge or server
UpdateBadge(Z_ID,K,Z,ID)
516
BadgeDetected
518
InitData(Kin, Kout, IDlist)
514
AccessUpdate(Z_ID,K,Z,ID)
517
AccessInvite(Zto)
519
520
Freeze(RT1) StartTimer(RT2)
RT2 Timer covering absence of badge feedback
Transient State: Waiting for badge feedback
521
522
TimeOut(RT2)
AccessRequest(ID, IDto, K)
524
Unfreeze(RT1)
523
StopTimer(RT2)
525
526
K = Kin ?
NO
AccessDenied
527
528
Intrusion(ID, Zin, Zout)
Warn Badge Holder
531
YES
529
InvalidAccess
NO
IDto found in IDlist ?
YES
AccessGranted(Kout,Tout)
532
530
OpenGate
Passage(IDto, Zin, Zout)
533
534

FIG 5

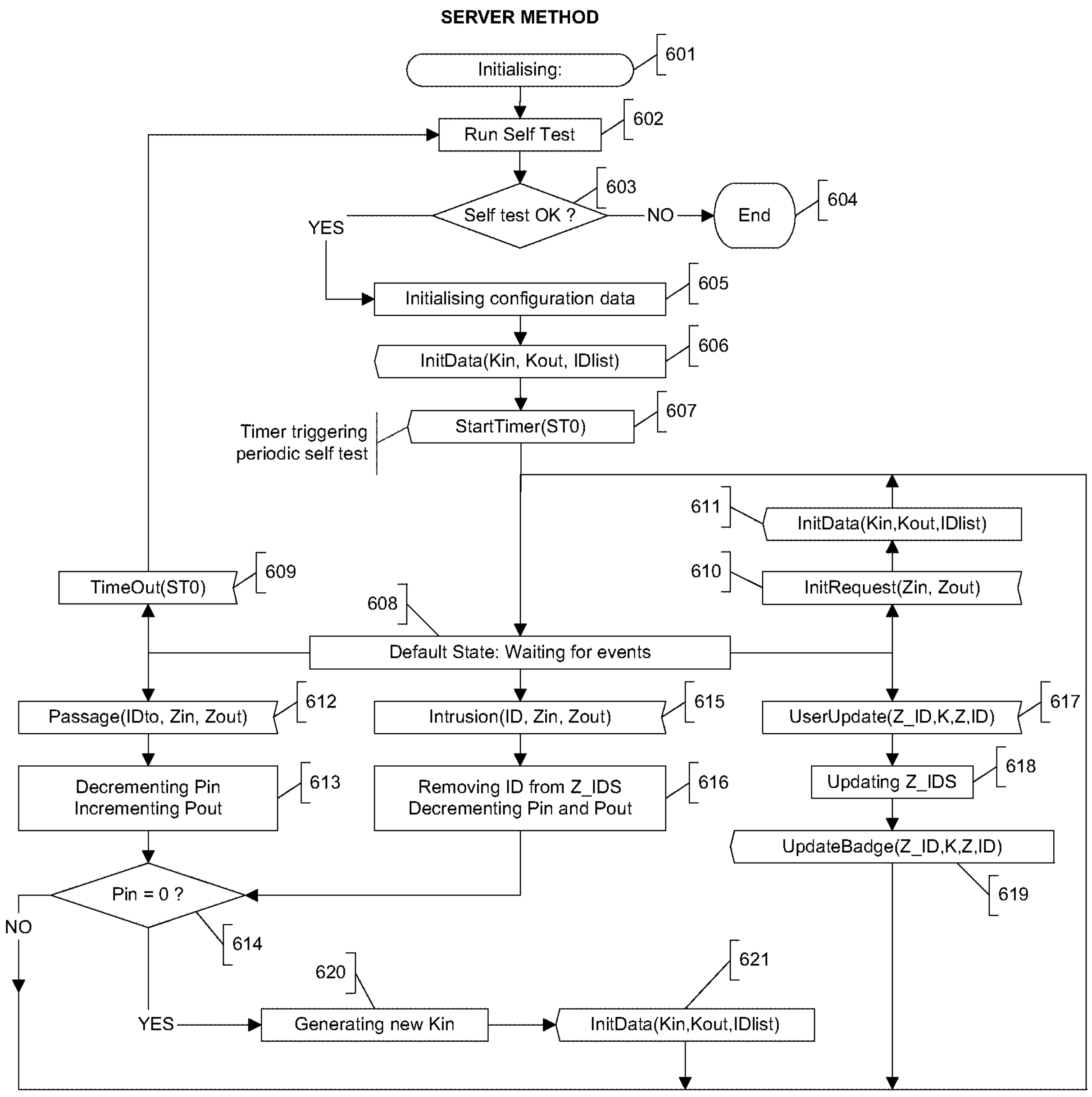

SERVER METHOD
Initialising:
601
Run Self Test
602
Self test OK ?
603
NO
End
604
YES
Initialising configuration data
605
InitData(Kin, Kout, IDlist)
606
StartTimer(ST0)
607
Timer triggering periodic self test
611
InitData(Kin,Kout,IDlist)
610
InitRequest(Zin, Zout)
TimeOut(ST0)
609
608
Default State: Waiting for events
Passage(IDto, Zin, Zout)
612
Intrusion(ID, Zin, Zout)
615
UserUpdate(Z_ID,K,Z,ID)
617
Decrementing Pin Incrementing Pout
613
Removing ID from Z_IDS Decrementing Pin and Pout
616
Updating Z_IDS
618
UpdateBadge(Z_ID,K,Z,ID)
619
Pin = 0 ?
NO
614
620
621
YES
Generating new Kin
InitData(Kin,Kout,IDlist)

FIG 6

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2006/066367

A. CLASSIFICATION OF SUBJECT MATTER
INV. G07C9/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G07C

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5 541 585 A (DUHAME DEAN C [US] ET AL) 30 July 1996 (1996-07-30) abstract; claim 1; figures 1,3,4 column 3, line 64 - column 4, line 62 column 7, line 3 - line 20 -----	1-26
A	US 2005/091338 A1 (DE LA HUERGA CARLOS [US]) 28 April 2005 (2005-04-28) abstract; claims 1-4,15 paragraphs [0183], [0184] -----	1-26
A	DE 199 32 147 A1 (INSYS GES FUER MICROCONTROLLER [DE]) 25 January 2001 (2001-01-25) the whole document -----	1-26
	-/--	

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search 25 January 2007	Date of mailing of the international search report 01/02/2007
Name and mailing address of the ISA/ European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Authorized officer Rother, Stefan

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2006/066367

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2005/083171 A1 (HAMILTON SHARON [US]) 21 April 2005 (2005-04-21) abstract; claim 1; figure 1 paragraphs [0015], [0109], [0110]	1-26
A	US 2004/138535 A1 (OGILVIE JOHN W L [US]) 15 July 2004 (2004-07-15) abstract; figure 1 paragraphs [0124] - [0152]	1-26
A	WO 03/060833 A1 (HILL ROM SERVICES INC [US]) 24 July 2003 (2003-07-24) abstract; claim 1; figures page 3, line 28 - page 7, line 30	1-26
A	US 2004/183682 A1 (TENARVITZ HENRY J [US]) 23 September 2004 (2004-09-23) abstract; claims 1,4,11,15; figures 1,2 paragraphs [0029] - [0056]	1-26
A	US 2003/001722 A1 (SMITH MARK T [US]) 2 January 2003 (2003-01-02) paragraphs [0036], [0038], [0043] - [0046]; claim 1; figures	1-26
A	US 2004/230488 A1 (BEENAU BLAYN W [US] ET AL) 18 November 2004 (2004-11-18)	
A	EP 1 513 084 A (LIECHTI AG [CH]) 9 March 2005 (2005-03-09)	

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2006/066367

Patent document cited in search report		Publication date	Patent family member(s)			Publication date
US 5541585	A	30-07-1996	NONE			
US 2005091338	A1	28-04-2005	NONE			
DE 19932147	A1	25-01-2001	NONE			
US 2005083171	A1	21-04-2005	NONE			
US 2004138535	A1	15-07-2004	NONE			
WO 03060833	A1	24-07-2003	AU	2003210479	A1	30-07-2003
US 2004183682	A1	23-09-2004	AU	2004222926	A1	07-10-2004
			BR	PI0409024	A	28-03-2006
			CA	2517955	A1	07-10-2004
			EP	1606761	A2	21-12-2005
			JP	2006520905	T	14-09-2006
			MX	PA05010075	A	27-06-2006
			WO	2004086287	A2	07-10-2004
US 2003001722	A1	02-01-2003	JP	2005527005	T	08-09-2005
			WO	03003333	A2	09-01-2003
US 2004230488	A1	18-11-2004	US	2005060233	A1	17-03-2005
			US	2005033686	A1	10-02-2005
			US	2005160003	A1	21-07-2005
EP 1513084	A	09-03-2005	AU	2004269444	A1	10-03-2005
			BR	PI0414060	A	24-10-2006
			CA	2537159	A1	10-03-2005
			WO	2005022421	A1	10-03-2005
			CN	1846222	A	11-10-2006
			IS	8393	A	31-03-2006