Regular Paper

Security Analysis on the Proactivized System against Latent Virus Attacks

Yuji Watanabe† and Hideki Imai†

The notion of proactive security of basic primitives and cryptosystems was introduced in order to tolerate a very strong "mobile adversary." However, even though proactive maintenance is employed, it is a hard problem to detect the viruses which are skillfully developed and latent in the memory of servers. We introduce a new type of virus attacks, called *latent virus attack*, in which viruses reside in the intruded server and wait for the chance for viruses colluding with each other to intrude more than the threshold of servers. The main subject of this paper is to analyze the resilience of proactive system against latent virus attacks and present how to enhance the security against such virus attacks. At first, the robustness of proactivized systems against this attack is estimated by probabilistic analysis. Next, in order to enhance the resilience against such virus attacks, we introduce *active rebooting aproach*, in which the reboot procedure on a predetermined number of servers is performed in the total independence of servers being infected or not. The effectiveness is also evaluated. As far as we know, this work is the first proposal for enhancing the robustness of proactive system against latent virus attacks.

1. Introduction

Threshold protocols [5)~7)] address a variety of adversaries and a variety of attacks. They maintain appropriate security against illegal hackers, insiders, disgruntled ex-employees, computer viruses, and other agents of data espionage and destruction.

Threshold protocols maintain secrecy in the face of up to $k-1$ adversaries and yet achieve data integrity and availability with the cooperation of k out of n shareholders. They are called "(k, n) *threshold protocol*," which is based on the presupposition that the ability of an attacker is less than a predetermined threshold.

However, it is substantially difficult to estimate the ability of an adversary quantitatively and to set a reasonable threshold, especially in case they are an attractive target for break-ins. For instance, we consider a $(3, 5)$ threshold protocol (in practice, MasterCard/Visa SET root key certification system has been implemented as a 3 out of 5 RSA threshold signature scheme [8)]). This protocol assures the security in a cryptographic sense, provided the adversary can corrupt (during the entire lifetime of the protocol) only at most 2 servers. It is natural that we should be concerned what the vulnerability of the system is, in other words, "what is the probability that the adversary can corrupt more than 3 servers?"

Moreover, in some services, such as certification authorities (CA), the system must remain secure for a very long period of time. For securing such a service, the system is composed of multiple servers which engage in a threshold protocol. However, in such systems, it is intuitively clear that it is difficult to restrict the attack within a constant fraction of servers during an entire lifetime of the system. (Given a sufficient amount of time, an adversary can break into servers one by one, thus eventually compromising the security of the system.)

A trivial but effective method to enhance the security against this fault is to set the threshold high enough to prevent the attack. Upper bounds of possible thresholds are therefore to be chosen as the proportion to the number of whole servers (typically the majority of servers), i.e., this method results in increasing the number of participating servers. Therefore, disadvantages of this approach include the complicated management of a server's security.

On the other hand, recently, the notion of "proactive security [1)]" of basic primitives and cryptosystems that are distributed amongst servers was introduced in order to tolerate a very strong "mobile adversary" without increasing the number of servers. This adversary may corrupt all servers throughout the lifetime of the system in a non-monotonic fashion (i.e., recoveries are possible) but the adversary is unable to compromise the secret if at any time period it does not break into more than $k-1$ loca-

† Institute of Industrial Science, University of Tokyo

tions ($k = \lfloor l/2 \rfloor$ is optimum [1]). Proactive security adds a periodic refreshing of the contents of the distributed servers' memories. Therefore, the knowledge of the mobile adversary (representing: hackers, viruses, bad administrator, etc.) obtained in the past from compromising at most $k-1$ number of servers is rendered useless for the future. As a result, the system can tolerate a "mobile adversary" which is allowed to potentially move among servers over time with the limitation that it can only control up to $k-1$ servers during a period of time [9]. This setting was originally presented by Ostrovsky and Yung [1]. The notion of "proactive security" assures increased security and availability of the cryptographic primitive [2)~4)].

1.1 Latent Virus Attack to Proactive System

We consider a very pragmatic scenario. Very diverse viruses are produced by malicious people and try various possible means to infect servers and to avoid the detection. Once the existence of a virus is detected by virus detection tools (e.g., anti-virus scanners) or checking protocols at run-time (e.g., VSS [10),11)]), the system instantly triggers the reboot operation of the infected server in order to remove the virus from the server completely [1),2)].

Computer viruses (as well as biological viruses) go through several processes (infection $\rightarrow$ latency $\rightarrow$ activation) until they cause damage to infected servers. Once the viruses become active and disturb the system in a malicious way, they are removable by checking the protocol or by detecting the infection. However, how can we detect and remove the virus in the latent period? Proactive maintenance provides the measure to find malicious behavior, but does not provide the measure to detect latency of a virus. Therefore, detection of a latent virus relies on virus detection tools. However, it is an intuitively hard problem to detect the viruses which are skillfully developed and latent in the memory of servers. Cohen [12] showed that a perfect defense against computer viruses is impossible. In fact, we never know perfect virus detection tools which detect all latent viruses without exception before their activation.

Furthermore, we modify the model of latent virus to a more powerful one which adaptively causes malicious corruption. In this model, latent viruses reside in the intruded server and wait for the chance for viruses colluding with each other to intrude more than the threshold of servers. Once the chance has come, all viruses become active and compromise the security of the system.

Of course, a latent virus setting seems to be rather theoretical but must be a potential threat, because we can easily see its reality by observing a vicious spiral of construction of virus detection tools and appearance of new type of stronger virus. A latent infection is a potential threat over many network-systems, but is hardly manageable by cryptographic techniques.

1.2 Our Results

The notion of proactive security was introduced in order to realize the long-term security against break-ins. A proactivized system includes the mechanism to refresh the servers' memories and renew the exposed shares into new one in order to make the old one useless for the adversary, as well as the mechanism to detect malicious behavior during the protocol.

However, proactive systems provide no measure to detect the latent viruses before their activation, so it depends on the ability of the virus detection tools. Therefore, a proactive system is not robust against the attack by latent viruses, due to the hardness to detect latent viruses which is produced skillfully.

The main purpose of this study is to show a method for enhancing the robustness of proactive protocol against latent virus attacks. Our contribution in this paper is as follows. At first, we estimate the robustness of proactivized systems against this attack by probabilistic analysis. As a result, we show that if the virus detection rate is higher than a certain threshold, it is possible for proactive maintenance to make the system robust, while, if less than the threshold, the failure probability of the system is dependent only on the virus infection rate.

In order to enhance the resilience against such virus attacks, we propose the notion of *active rebooting*, in which the system performs the reboot procedure on a predetermined number of servers in total independence of servers being infected or not. We estimate the security of proactive maintenance with active rebooting by extending the probabilistic model. As a result, we show that active rebooting enables us not only to enhance the security against the viruses with higher infection rate, but also to make the system robust even in the case of a low detection rate. Moreover, we show that it is effective

even in the case the number of servers which are forced to carry out the reboot operation every update phase is comparatively small.

1.3 Organization

The organization of this paper is as follows. In the next section, we explain the basic definitions and a probabilistic model of virus infection and detection in a proactive maintenance. In Section 3, we estimates the robustness of a proactivized system against latent virus attacks. Then, as an effective and practical countermeasure against such attacks, we show "active rebooting" method in Section 4. Finally, we shall conclude in Section 5 with a summary of our results and future works in this area.

2. Preliminaries

2.1 Model

We basically use the model by Herzberg, et al.[2] with a slight modification. This modification is due to simplification of the analysis on latent virus attacks (see Section 2.3).

In (k, n)-threshold protocol[13], a system of n servers $\mathcal{P} = \{P_1, \ldots, P_n\}$ share a function f_s for some key s. This secret value s is shared among n servers through a (k, n)-threshold scheme (i.e., s is divided into n shares $\{s_1, \ldots, s_n\}$ such that $k-1$ shares provide no information on the secret, while k shares suffice for the reconstruction of the secret). We say that the protocol is *a (k, n)-threshold protocol* if, when k uncorrupted servers are active, for any x, the shared function f_s can be reconstructed to compute $f_s(x)$ even in the presence of $k-1$ corrupted servers, yet nothing about $f_s(x')$ is revealed for $x' \neq x$.

(k, n)-proactive protocol[2] is a variant of (k, n)-threshold protocol in order to enhance the security against the adversary who can corrupt all servers throughout the entire lifetime of the system in a non-monotonic fashion, but can corrupt no more than $k-1$ out of n servers during any period of time. Proactive protocol works as follows (**Fig. 1**). Let t be the time which has passed from the beginning of the operation of servers. For instance, we assume a time unit is a day. The lifetime of the secret s is divided into periods of time (e.g., days, weeks, etc.). We define ξ as $\xi = \lfloor t/\tau \rfloor$ where τ is the length of single time period. Accordingly, if $(\xi-1)\tau \leq t < \xi\tau$, we say that the system at the time t belongs in *ξ-th time period*.

Each time period is divided into two parts, *operating phase* and *update phase*. The time length of them are denoted as τ_o and τ_u, respectively (i.e., $\tau = \tau_o + \tau_u$), where the update phase is negligibly short when compared to the length of a time period, that is, $\tau_u \ll \tau$, $\tau_o \approx \tau$. We refer to the period within $(\xi-1)\tau \leq t < (\xi-1)\tau + \tau_u$ as *ξ-th update phase* $\mathcal{U}^{(\xi)}$, and the period within $(\xi-1)\tau + \tau_u \leq t < \xi\tau$ as *ξ-th operating phase* $\mathcal{O}^{(\xi)}$. The largest part of each time period belongs to the operating phase and only a short period at the beginning of each time period belongs to the update phase.

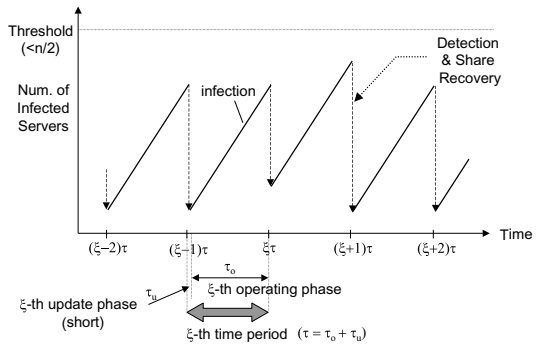

Fig. 1 A diagram illustrating proactive maintenance.

At update phases, the servers engage in an interactive update protocol, after which they hold completely new shares of the same secret. A non-faulty majority of servers trigger a reboot operation of faulty servers during an update phase, in order to bring a completely fresh version of the program from ROM to memory. We refer these operations during update phases as *proactive maintenance*.

Note that a server which has been infected by the virus but has not detected it yet does not perform the reboot operation. Accordingly, the viruses can stay in servers' memories as long as it is not detected. Considering the behavior of such viruses is the main subject of this paper.

For the simplicity of analysis, we do not consider the corruption during an update phase. Even if a server is corrupted during an update phase, we can consider the servers as corrupted during both periods adjacent to that update phase. It is also not a realistic concern in our setting, where the update phase is negligibly short when compared to the length of a time period.

On a removal of viruses through reboot procedures, we assume that the adversary intruding the servers $\mathcal{P}$ is "*removable*" in the sense that all of the shares are thrown away and new shares of the secret are created. These was car-

ried out through a *reboot* procedure when it is detected by explicit mechanisms by which a majority of (honest) servers always detects and alerts about a misbehaving server or by regular detection mechanisms (e.g., anti-virus scanners) available to the system management.

Triggering the reboot operation of a misbehaving server relies on the system management which gets input from a majority of (honest) servers. Once the mechanism is found to be infected, a complete reboot is performed in order to bring a completely fresh version of the program from ROM to memory. In this paper, we assume that viruses cannot survive a physical reboot of the machine (for its detail, see Ref. 2)), and for this assumption, we also assume each server has the minimum amount of trusted hardware for I/O and ROM.

2.2 Virus Intrusion Detection Model

One of the purposes of this paper is to estimate the robustness of the system based on a threshold structure against virus attacks.

Generally, at first, viruses attempt to intrude into a server's memory. After they are latent within a certain period of time after their intrusion, they act and behave in various malicious ways. However, once such viruses act, they are detectable with extremely high probability by the virus detection mechanism of the system. Accordingly, the powerful attacker who attempts to corrupt the system totally design viruses which are adaptively latent until intruding more than a threshold of servers. Therefore, we consider adaptive viral behavior such that viruses do not act until they succeed in intruding and lurking into more than a threshold of servers.

According to these observations, the state of an individual server can be placed in the following three categories. Let $\mathcal{C}(t)$ be a set of servers which are uninfected by viruses at the time t, $\mathcal{I}(t)$ be a set of servers which are infected by viruses but not aware of their own infection, and $\mathcal{D}(t)$ be a set of servers which are infected by viruses and have already detected the infection of viruses in their own memories. Define $n_c(t) = |\mathcal{C}(t)|$, $n_i(t) = |\mathcal{I}(t)|$ and $n_d(t) = |\mathcal{D}(t)|$, where $n = n_c(t) + n_i(t) + n_d(t)$.

Figure 2 shows the simple model of state transition with respect to virus infection and detection. Let β, called *infection rate*, be the probability with which an individual uninfected server is infected by viruses in a unit time, i.e., for $\forall \xi,\ \forall t \in \mathcal{O}^{(\xi)},\ \forall P_{j'} \in \mathcal{C}(t)$,

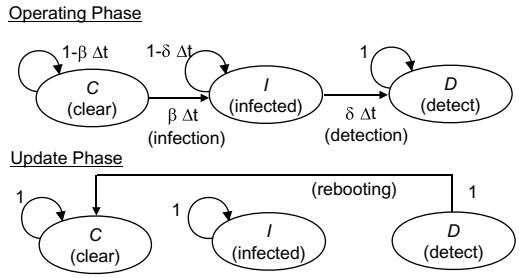

Fig. 2 A model of virus infection and detection on a proactive system.

$$\Pr[P_{j'} \in \mathcal{I}(t + \Delta t)] = \beta \Delta t$$
$$\Pr[P_{j'} \in \mathcal{C}(t + \Delta t)] = 1 - \beta \Delta t.$$

After a virus intruding and being latent in $P_{j'} \in \mathcal{C}(t)$, $P_{j'}$ belongs to $\mathcal{I}(t + \Delta t)$. Consequently, $P_{j'} \in \mathcal{I}(t + \Delta t)$, $n_c(t + \Delta t) = n_c(t) - 1$ and $n_i(t + \Delta t) = n_i(t) + 1$.

Proactivized servers should take enough measure to prevent a virus infection from propagating successively by strict management of communication among servers or use of diverse operating systems among each other. If not, it may be feasible for the number of infected servers to exceed the predetermined threshold. So, we do not consider the successive propagation of virus infection but consider the simple model that the infection rate β is invariable throughout the lifetime of the system.

Moreover, most of a proactivized system is symmetrically constructed, which means that there is no difference among the roles the different servers play. So, we use the model that each individual server is infected by viruses at the same rate β among each other.

Similarly, let δ, called *detection rate*, be the probability with which an individual infected server become aware of viruses intruding and being latent in its inside in a unit time, i.e., for $\forall \xi,\ \forall t \in \mathcal{O}^{(\xi)},\ \forall P_{j'} \in \mathcal{I}(t)$,

$$\Pr[P_{j'} \in \mathcal{D}(t + \Delta t)] = \delta \Delta t$$
$$\Pr[P_{j'} \in \mathcal{I}(t + \Delta t)] = 1 - \delta \Delta t$$

After $P_{j'} \in \mathcal{I}(t)$ detected the infection of virus, $P_{j'}$ belongs to $\mathcal{D}(t + \Delta t)$. Consequently, $P_{j'} \in \mathcal{D}(t + \Delta t)$, $n_i(t + \Delta t) = n_i(t) - 1$ and $n_d(t + \Delta t) = n_d(t) + 1$. Viruses detected in $\mathcal{O}^{(\xi)}$ are surely removed from server's memory as a result that the reboot procedure is triggered with probability 1 in the update phase $\mathcal{U}^{(\xi+1)}$. (This setting is well defined in Ref. 2).)

At each update phase, the system performs proactive maintenance in order to renew old shares of all servers into new ones. The update phase is negligibly short when compared to the

length of a time period, that is, $\tau_u \ll \tau$. Therefore, we assume that there is not any new infection and detection within an update phase, because it is reasonable that the probability of occurrence of new infection and detection within an update phase is negligibly small compared with that within an operation phase.

We consider in this paper the probability of occurrence of failure caused by latent viruses intruding into servers. Accordingly, we define the failure of (k, n)-threshold (proactive) protocol as follows.

Definition 2.1 (Failure probability) For a (k, n) threshold (proactive) protocol, we say that the system *fails* if the number of corrupted (or infected) servers exceeds $k-1$ at some point of time t. $P_f(t)$ denote the probability of occurrence of this failure from the beginning of the system till the time t. We call $P_f(t)$ the *failure probability*, which can be defined as

$$P_f(t) := \Pr[\exists \zeta < t,\ n_i(\zeta) + n_d(\zeta) \geq k].$$

2.3 Latent Virus Attack

At the end of this section, we informally define the adversary that we consider in this paper. Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ denote a system of n servers employing the (k, n)-proactive protocol. We call a group of adversary $\mathcal{A} = \{A_1, \ldots, A_n\}$ *latent viruses* if $\mathcal{A}$ meets the following properties.

- A_j can infect server P_j in a unit time at the rate β.
- After the infection, A_j is latent in server's memory without carrying out malicious behavior and wait for the instruction from the director.
- Once more than a threshold of servers have been infected by a subset of $\mathcal{A}$, denoted as $\mathcal{A}'$, $\mathcal{A}'$ starts to corrupt intruded servers simultaneously in arbitrary malicious way, which may lead to disrupt the proactive protocol.
- $\mathcal{A}$ is computationally bounded and therefore can not break any of the underlying cryptographic primitives used.

Note that $\mathcal{A}$ can wait for the opportunity to disturb the servers, since he knows whether the intrusion of viruses into servers results in success or failure. It is intuitively reasonable that viruses for intruding into proactivized servers are more adaptive and intellectual than biological viruses epidemic or the normal computer viruses.

We will take up the $(4, 7)$-proactive protocol as a typical example. Our greatest concern in this paper is to enhance the security against latent virus attack without modifying the existing system, so we does not leave increasing the number of servers out of consideration. (Of course, it is one of the most effective solutions. However, it is a trivial fact that increasing the number of participating servers results in making the system more robust against the failure.) Narrowing an argument down to the case of a certain number of servers (here we take up 7 servers' case), we evaluate its security against a latent virus attack and show how to enhance the security without increasing the number of participating servers.

Throughout the following analysis, we assume the length of the time period τ is a day. So, β and δ mean the probability of occurance of new infection and detection within a day, respectively.

3. Analysis

3.1 Robustness of Latent Virus Attacks (Unproactivized Systems)

At first, we examine a failure probability of (k, n)-threshold protocol without proactive maintenance against latent virus attacks. In order to account for the relationship between obscure parameters, we use the probabilistic approximation of virus infection and detection. (This description was introduced by Cohen [12].) Accordingly, we do not deal with states of servers at any point of time as deterministic states but as probabilistic states by representing the probability of transition from one state to another.

Note that there are no transition in the direction of decreasing the number of infected servers, because the information of shares which has been leaked out by viruses cannot be recovered without being refreshed into new shares by proactive maintenance, even if viruses are detected and removed from servers' memories. Consequently, the security of (k, n)-threshold protocol against latent virus attacks can be equivalent to that of (k, n)-proactive protocol in case of $\delta = 0$. Therefore, the result of (k, n)-threshold protocol (without proactive maintenance) is an upper bound of a failure probability against this attack.

A typical threshold is $k = \lfloor n/2 \rfloor$ in most of (k, n)-threshold protocols, because this guarantees the existence of k honest servers and the corruption of no more than $k - 1$ servers (for details, see Ref. 1)). Throughout this paper, we

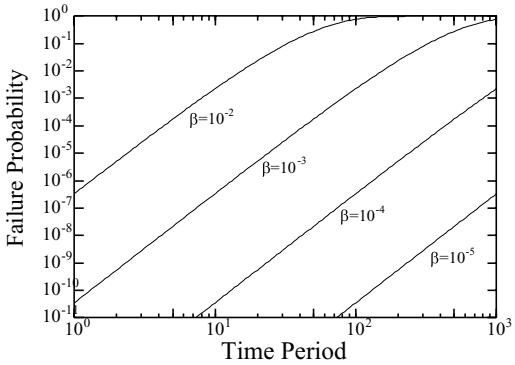

Fig. 3 A failure probability of the $(4, 7)$-threshold protocol.

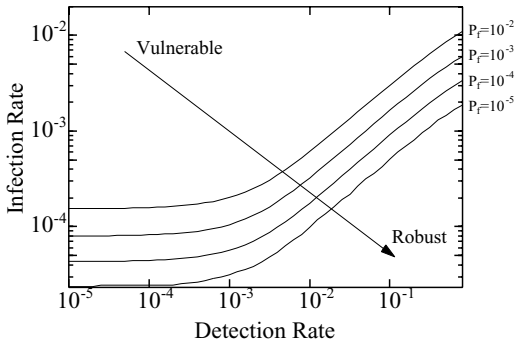

Fig. 4 Virus infection rate against virus detection rate achieving the failure probability $10^{-2}, 10^{-3}, 10^{-4}$ and 10^{-5} after 1000 time periods.

assume $k = \lfloor n/2 \rfloor$.

Figure 3 shows the failure probability of $(4, 7)$-threshold protocol. The horizontal axis denotes the progress of time. As one can see clearly, the failure probability of the system increases along with the progress of time. Let us define η as the time when the failure probability $P_f(\eta)$ exceeds $1/2$ at first, which is given as $\eta = \ln 2/\beta \approx \beta^{-1}$.

Let us consider after 1000 $(= 10^3)$ time periods, for example (after about 3 years from the beginning of the system). If $\beta = 10^{-2}$ or $\beta = 10^{-3}$, there is no doubt the system will fail. Even if $\beta = 10^{-4}$, the system is not secure enough to ensure long-term security. It is required that the virus infection rate should not exceed 10^{-5} as long as the system should be available for more than 3 years (in case of the $(4, 7)$-threshold protocol without proactive maintenance).

3.2 Robustness against Latent Virus Attacks (Proactive Systems)

As figure 3 shows, the assumption of threshold does not hold over a long period of time but within a certain period of time. In threshold protocols without proactive maintenance, once the secret information in servers has leaked out by the viruses, there is no measure to refresh the information and to make the leaked information useless. So, given a sufficient amount of time, the viruses can break into the servers one by one, thus eventually compromising the security of the system. Accordingly, the attacks of the viruses with high infection rate are serious concern in systems that must remain secure for long periods of time.

Proactive maintenance is highly effective against such viruses, due to the mechanism to remove detected viruses and to renew old shares into completely new ones (see Fig. 1). Therefore, a proactive protocol is more robust against latent virus attacks than a threshold protocol without a proactive maintenance. Actually, it is an interesting question how the proactive maintenance enhances the security against the latent viruses with infection rate $\beta = 10^{-2} - 10^{-4}$ which are a threat to $(4, 7)$-threshold protocol without proactive maintenance as mentioned in Section 3.1.

We estimate the above probabilistic model of virus infection and detection in a proactive maintenance. Let us define the probability function $S(j_1, j_2, t)$ for $\forall \xi, \forall t \in \mathcal{O}^{(\xi)}$ as

$S(j_1, j_2, t) := \Pr[n_i(t) = j_2 - j_1, n_d(t) = j_1]$,

where $n - j_2$ servers are uninfected by viruses. $S(j_1, j_2, t)$ means the probabilistic state of the system with respect to virus infection and detection. We can easily see, for $\forall \xi, \forall t \in \mathcal{O}^{(\xi)}$

$$P_f(t) = 1 - \sum_{j_2=0}^{k-1} \sum_{j_1=0}^{j_2} S(j_1, j_2, t)$$

Now we estimate the security of the system which is engaged in a $(4, 7)$-proactive maintenance after 1000 time periods. **Figure 4** shows the relationship between virus infection rate β and detection rate δ achieving each failure probability P_f after 1000 time periods. If the point (δ, β) is located above each of the boundary lines, the system cannot keep the corresponding security against latent virus attacks, otherwise, the system assures the security with less failure probability.

There are remarkable differences on the shape of all lines between $\delta > \delta_{th}$ and $\delta < \delta_{th}$, where δ_{th} is the lower bound of effective detection of viruses and in this case (i.e., $(4, 7)$-proactive protocol), we can find $\delta_{th} \simeq 10^{-3}$. From this

observation, we can easily see the following requirements for keeping the system secure.

If $\delta > \delta_{th}$, the requirement for achieving an failure probability P_f is $\beta/\delta < \rho_{th}$, where ρ_{th} is the threshold of infection/detection raito which is determined by the number of servers and the failure probability to be achieved. In this case, the detection and removal of viruses by proactive maintenance works for enhancing security against latent virus attacks. If $\delta < \delta_{th}$, the requirement for achieving an failure probability P_f is $\beta < \beta_{th}$, where β_{th} is determined by the number of servers and the failure probability to be achieved. In this case, there is hardly any contribution of proactive maintenance to enhance the security against latent virus attacks because of the lack of the ability to detect the viruses. Therefore, the failure probability of the system practically depends on the infection rate β.

Accordingly, the security of the system against latent virus attacks depends heavily on the ability to detect the infection of the viruses. Especially, in case of $\delta < \delta_{th}$, we can find that the detection and removal of viruses in proactive maintenance does not work effectively enough. This threshold δ_{th} also exists in the case of n servers in general, because the proactive protocol in case of the low detection rate is almost equivalent to unproactivized protocol in which the failure probability does not depend on the total number of servers (see Section 3.1) Increasing the detection rate can be achieved by improving the virus detection mechanism (anti-virus scanner, etc.), this parameter is too unknown and fluid to estimate the security of the system appropriately. Therefore, from the viewpoint of constructing a mission critical system, it is to be desired that we could cope with more powerful viruses by more controllable means even in a case where the ability to detect viruses is not enough. The method to meet these demands is "active rebooting method" which we present in the following session.

4. Active Rebooting

Above proactive maintenance does not work enough in case of low detection rate. This is caused by the fact that the server is not rebooted until it becomes aware of the existence of viruses (passive rebooting). Therefore, the unawareness of viruses intruding and being latent in servers results in neglecting to take appropriate measures promptly. Accordingly, we propose a new method of rebooting servers, which we call "*active rebooting*" method. Active rebooting assures us that in each update phase, more than a predetermined number α of servers (of course, $\alpha < k$) is rebooted and their states were made uninfected. Since active rebooting method does not depend on detection of viruses for their removal, it is expected to remarkably reduce the failure probability especially in case of low detection rate.

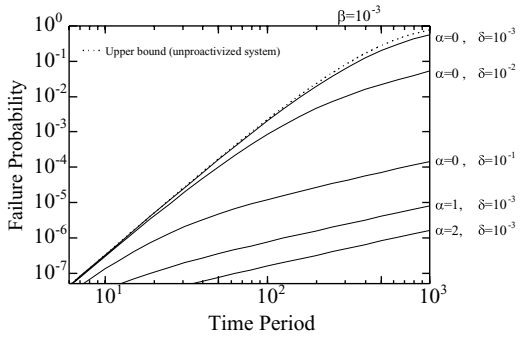

Fig. 5 A failure probability with respect to the progress of time for $\beta = 10^{-3}$.

In the same way as the previous section, we estimate the security of the system which is engaged in a $(4, 7)$-proactive maintenance with active rebooting. **Figure 5** shows the failure probability $P_f(t)$ with respect to the progress of time for $\beta = 10^{-3}$. The vertical axis means the failure probability and the horizontal axis means the progress of time. A dotted line is the upper bound of failure probability of the $(4, 7)$-threshold protocol (see Section 3.1). If the detection rate $\delta < 10^{-2}$, there is no improvement even though the system performs the proactive maintenance. Applying active rebooting to update procedure enables us to reduce the failure probability even in case of low detection ability. We remark that active rebooting is effective as compared with passive rebooting even when small α, that is, the increase of processing is comparatively small.

Figure 6 shows the relationship between the infection rate β and the detection rate δ achieving the failure probability $P_f = 10^{-3}$ after 1000 time periods. If the point (δ, β) is located above each of the boundary lines, the system cannot keep the corresponding security against latent virus attacks, otherwise, the system assures the security with less failure probability. The curve of $\alpha = 0$ (without active rebooting) is the same as Fig. 4.

Employing the active rebooting enhances ro-

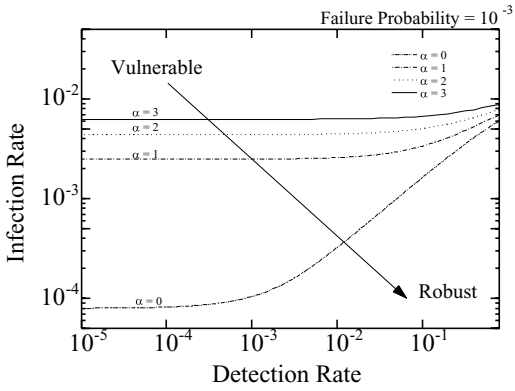

Fig. 6 Virus infection rate against virus detection rate achieving the failure probability 10^{-3} after 1000 time periods.

bustness against latent virus attack with higher infection rate β throughout the entire range of δ, so that the failure probability is not so dependent on the detection rate δ. We mention that the proactivized system with active rebooting even in the case of relatively small δ can assure the same level of security against the latent viruses as that of high δ. This is caused by the mechanism of active rebooting removing infected and hidden viruses in servers. In the case of n servers, the size of α is also small compared with the size of n, because the removal rate of viruses by active rebooting can be estimated as α/n. This result means that the use of active rebooting is highly effective against the attack of latent viruses, even though β and δ may not be measured quantitatively in practice. Therefore, it is important to use proactivized system in an active rebooting manner in order to maintain the security against unknown viruses.

5. Conclusions

In this paper, we proposed the method for enhancing the robustness of proactive protocol against latent virus attacks. Our contribution in this paper is as follows.

At first, we estimated the robustness of proactivized systems against this attack by probabilistic analysis. Next, in order to enhance the resilience against such virus attacks, we proposed the notion of *active rebooting*, in which the system performs the reboot procedure on a predetermined number of servers in the total independence of servers being infected or not. We estimated the security of proactive maintenance with active rebooting by extending the probabilistic model of proactive maintenance. As far as we know, this work is the first proposal for enhancing the robustness of a proactive system against latent virus attacks.

References

1) Ostrovsky, R. and Yung, M.: How to withstand mobile virus attacks, *Proc. PODC'91*, pp.51–59 (1991).
2) Herzberg, A., Jarecki, S., Krawczyk, H. and Yung, M.: Proactive secret sharing, or How to cope with perpetual leakage, *Proc. CRYPTO'95*, pp.339–352 (1995).
3) Herzberg, A., Jakobsson, M., Jarecki, S. and Krawczyk, H.: Proactive public key and signature systems, *Proc. 4th ACM Symposium on Computer and Communication Security'97* (1997).
4) Frankel, Y. Gemmell, P., Mackenzie, P. and Yung, M.: Proactive RSA, *Proc. CRYPTO'97*, pp.440–454 (1997).
5) Shamir, A: How to share a secret, *Comm. ACM*, Vol.22, pp.612–613 (1979).
6) Desmedt, Y.: Threshold cryptosystem, *European Trans. Telecommunications*, Vol.5, No.4, pp.449–457 (1994).
7) Gemmell, P.S.: An introduction to threshold cryptography, *CryptoBytes*, Vol.2, No.3, pp.7–12 (1997).
8) Frankel, Y. and Yung, M.: Distributed public key cryptography, *Proc. PKC'98*, pp.1–13 (1998).
9) Canetti, R., Gennaro, R., Herzberg, A. and Naor, D.: Proactive security: Long-term protection against break-ins, *CryptoBytes*, Vol.3, No.1, pp.1–8 (1997).
10) Feldman, P.: A practical scheme for non-interactive verifiable secret sharing, *Proc. FOCS'87*, pp.427–437 (1987).
11) Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing, *Proc. CRYPTO'91*, pp.129–140 (1991).
12) Cohen, F.: Computer viruses, theory and experiments, *Computers & Security*, Vol.6, pp.22–35 (1987).
13) De Santis, A., Desmedt, Y., Frankel, Y. and Yung, M.: How to share a function securely, *Proc. STOC'94*, pp.522–533 (1994).
14) Adleman, L.: Abstract theory of computer viruses, *Proc. CRYPTO'88*, pp.354–374 (1988).
15) Watanabe, Y. and Imai, H.: Active rebooting method for proactivized system: How to enhance the security against latent virus attacks, *Proc. 1999 International Information Security Workshop* (*ISW'99*), pp.118–135 (1999).
16) Watanabe, Y. and Imai, H.: Probabilistic analysis on proactive security against latent virus attacks, Technical Report of IEICE,

ISEC99-36, pp.77–84 (1999).
(Received November 30, 1999)
(Accepted June 1, 2000)

Yuji Watanabe was born in 1973. He received his B.E. and M.E. degrees in information and communication engineering from University of Tokyo in 1996 and 1998, repectively. From 1998, He is currently working toward the Ph.D. degree in information and communication engineering in the University of Tokyo. He is a member of ACM and IACR.

Hideki Imai was born in Shimane, Japan on May 31, 1943. He received the B.E., M.E., and Ph.D. degrees in electrical engineering from the University of Tokyo in 1966, 1968, 1971, respectively. From 1971 to 1992 he was on the faculty of Yokohama National University. In 1992 he joined the faculty of the University of Tokyo, where he is currently a Full Professor in the Institute of Industrial Science. His current research interests include information theory, coding theory, cryptography, spread spectrum systems and their applications. He is an IEEE Fellow.