A Logical Model for Detecting Irregular Actions in Physical Access Environment

António Leong , Simon Fong, Zhuang Yan
Faculty of Science and Technology
University of Macau
{ccfong, syz}@umac.mo

***Abstract*— This paper proposes a framework to learn about irregular and abnormal behavior of access card user's pattern based on the real-time analysis data. The aim of the technique is to overcome the security level problem set by human and therefore relying on the collected real-time data in order to study the pattern of the access users. The concept of studying the irregular bahavior of the access user is using Predicate Logics for rule checking. An architecture of the model is described which includes the relationship between the user-level and the dependants as well as the accessing points.**

***Keywords*—Smart-card, Access Control, Intrusion Detection**

I. INTRODUCTION

Unlike logical access control systems or networks, most of the current physical access systems rely on door locking mechanisms for security. Based on some access control lists, which are sets of predefined rules and constraints used to validate the access permission of cardholders, the system via the hardware devices will unlock a door should the authorization is granted.

Such control-lists usually store security policies of individual users. By making use of public key cryptography, certificates containing security policies and access information of the cardholder can be stored safely within the card itself. The lists are usually updated periodically within a certain period of time, which might reduce security. Meanwhile if cardholders exhibit irregular card access behaviors, a real-time detection system can aid stopping fraudulent actions.

We developed a logical model for detection of irregular and irregular actions. A list of irregular behaviors can be identified a-priori based on empirical analysis and predicted using data-mining techniques. A logical mathematical model is built to formulate a knowledge base of irregular actions. This model is based on intuition and experience of field experts in order to select relevant statistical measures for anomaly detection. Based on this model a real-time irregular action detection framework is established. Irregular behaviors can be detected early that facilitate immediate remedies.

In this framework, door access devices are linked to a centralized server, allowing real time analysis of access pattern and information of the user's latest location within a closed environment.

II. ASSUMPTIONS, NOTATIONS AND DEFINITIONS

A. Participants

Table 1. Participants and roles.

Participant	Role
Issuer	Creates, signs and issues a card
User	Use the card to identify themselves in order to grant access permission to requested facilities or services
Relying party	Devices responsible for authentication and validation of card
Security officer	Receive security alert message and react according to defined security policies

B. Assumptions

The following assumptions are made for the model:

1. The relying party validates one user's card at a time.
2. The relying parties are installed on both sides of a door.
3. When authorization is granted to a user, only this user is legible to access the door (i.e. no shoulder-surfing).

C. Access points

Access points are defined as objects where users present their cards to the relying party for authentication and validation. The relying party has to authenticate the card and perform a set of validation checks according to the defined security check policy before authorization is granted.

For administration and management purposes, we assume that access points can be grouped according to their geographical distribution and ownership.

For our model, we first define objects in a logical form. We denote objects as access elements ae_{dn} belonging to a set AE, and security check policies are assigned to each of these elements. The subscript d is a descriptor used to represent the type of the access element and n is a sequential integer number that uniquely identify individual elements of each type. The following table defines some typical descriptors.

Table 2. Access points descriptors

Descriptor	Description
i	Elements located on entrances
o	Elements located on exits
r	Root elements
z	Constrained zone elements

Access group $ag_{pi}\Lambda\{ag_{kj}\}_n$ is defined as one type of access elements belonging to *AG*, which is a subset of *AE*. The subscripts *p* and *k* denote the type of access group, while *i* and *j* are integer numbers that uniquely identifies each group of the same type. Access group relationship can be represented as a graph, where nodes are individual access points or groups, or sets of access points and groups. An arc directed from ae_{dn} indicates that it belongs to group ag_{ki}. See Fig. 1.

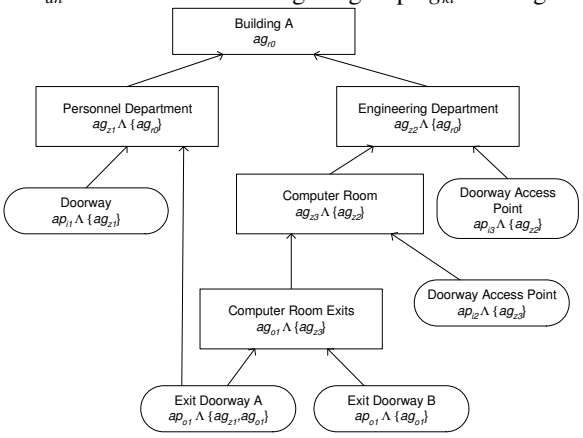


Fig. 1. An example of access group-hierarchy

Consider the example shown in Fig. 2. We define ag_{zx} as the group of all elements located within the constrained zone *zx,* which is a physically closed area with a set of *n* entrances $\{ap_{ik}\}_n$ and a set of *n* exits $\{ag_{om}\}_n\Lambda\{ag_{zx}\}$.

As defined in our assumption, it is logically not possible to access any elements belonging to group ag_{zx} unless the user is located physically within zone *zx*. Users must have the rights for admittance to enter the room through the entrances $\{ap_{ik}\}_n$ before accessing any of the elements belonging to group ag_{zx}.

Expressed mathematically, if the access is a feasible one, at least the condition $\forall ae_{pi}\in AE, al(uz)\in\{ap_{ik}\}_n \rightarrow ac(uz)\in ag_{zx}$ must be satisfied, where *al*(*uz*) and *ac*(*uz*) are last and current access points, respectively. We denote this property as constrained access sequence and we use operator Ω to denote this relationship. The notation to $ae_{pj}\Omega ap_{ki}$ is used to denote that an element ae_{pj}, either an access point or group, can only be activated if and only if the point ap_{ki} has been accessed.

Using our notation, we can define the elements located within the constrained zone *zx* as $ag_{zx}\Omega\{ap_{ik}\}_n$, and due to the inheritance property, all exits located inside *zx* $\{ag_{om}\}_n\Omega\{ap_{ik}\}_n$ can only be activated after succeed permission to enter through the entrances $\{ap_{ik}\}_n$.

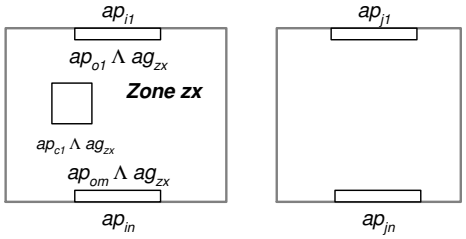


Fig. 2. Constrained access

For all points not belonging to $ag_{zx,}$ they must be activated only after any of $\{ap_{om}\}_n$ is accessed, i.e., the users must exit physically from zone *zx* before they can go to any other points. Using our notation, the access is eligible when the following condition is met $\forall ae_{pi}\in AE, al(uz)\in ag_{zx}\wedge ac(uz)\notin ag_{zx}\rightarrow ae_{pi}\Omega\{ap_{om}\}_n$.

D. Users

Access permissions are determined by a set of security rules defined in the user permissions policy, which describes the type of accesses and constraints that users are allowed or limited over each of the access points.

We define the set of all user elements as *ue*, belonging to the set *UE*. User permissions policies are assigned to each of these elements. User elements, which are composed by the set of users and groups, can belong to one or more groups. The operator is used to denote that user element $ue_{pi,}$ which can be either a single user or a collection of users, belongs to the groups defined in the vector of size *n,* $\{ug_{kj}\}$. This can be represented as $ue_{pi}\Lambda\{ug_{kj}\}_n$, where the subscripts *p* and *k* are descriptors used to characterize the type of the user. We assume that defined in the ue_{pi} directly inherits all user permissions policies from its parents defined in the vector $\{ug_{kj}\}_n$, and individual elements can override the policies defined by their parents and they can also define new policies. User group-hierarchy can be also represented as a graph, as shown by the example in Fig. . To avoid acyclic relationships, if an element ue_{pi} belongs to another element ue_{kj}, then ue_{pi} cannot be a member of ue_{kj}. Users are defined as *uz* belonging to *UZ*, a subset of *UE*. User group defines a collection of users, which is defined as $ug\in UG\subset UE$. The notation ug_{r0} is used to define the root user group.

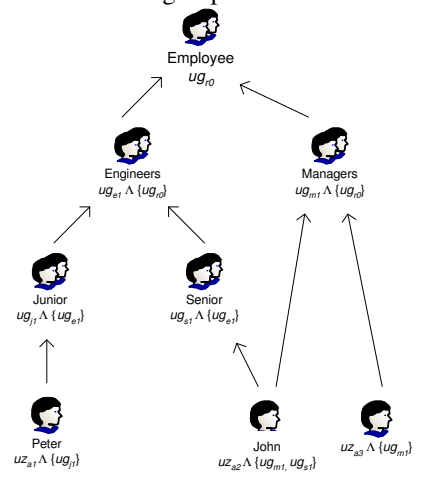


Fig. 3. An example of user group-hierarchy

III. Logical Model

Here we construct a logical model that allows computerized systems to interpret irregular card usages and return a quantitative alert level that represents the risks derived from it. Based on our model, we define a set of empirical formulae.

A. Definition

The granted-or-denial rules in security policies are usually predefined by security officer. They may not be sufficient to detect irregular access behavioral patterns that occur in real-time basis. By generalizing these static rules mathematically, we define the violation function $vio(\{v_c\})$ as follows:

$$vio(\{v_c\}) = c(\{v_c\}) \quad \text{Equation 1}$$

where c is conditional function, $\{v_c\}$ is input vector of conditions factors. The violation function performs a set of checks according to the security policy and returns a Boolean value. The Boolean results are represented as 0 and 1 that respectively corresponds to the approval or denial of the detection verdict of critical violations. The return values of $vio(\{v_c\})$ are calculated based on the conditional function $c(\{v_c\})$, which represents the validations rules in mathematical form. As input parameters vary according to different type of checks and validations, we use an vector $\{v_c\}$ to generalize the set of input parameters. We define $\phi(\{cond\})$ as Boolean conversion function that takes the generalized expression $\{cond\}$ as input and converts the Boolean results of the logical relationship defined by $\{cond\}$ to either 0 and 1, corresponding to the results of false and true respectively. Consider the example $\phi(a < b)$, where a and b are positive integer numbers. If a is smaller than b, then $\phi(a < b)$ returns a value of 1, else it returns a value of 0.

The conditional function $c(\{v_c\})$ takes as input a set of validation parameters using the vector $\{v_c\}$. The result of function c is obtained by multiplying the return values of a set of logical and relational functions defined using the Boolean conversion function ϕ. Consequently, function *vio* will return a value of 1, corresponding to the condition of "no violations detected", or a value of 1 meaning that "violations detected".

Validations perform using Equation 1 will simply return a Boolean result represented by the values 0 and 1. They do not take account of dynamic changes in the real-time access control system and cannot distinguish some possible type of actions that the model might consider as abnormal or irregular. To handle this type of behaviors, we will develop a generalized model extended from Equation 1 and introduce the concept of security credits *s*, where $s \in S$. S is the set of security points defined in of real number set ranging from values $[s_{min}.s_{max}]$, defined by the security officer. The security credit value is set to s_{max} and its value is stored in the card when it is issued. During the validation process, if any type of abnormal action is detected, the value stored in the card $s_{card} \in S$ will be deducted by the amount of security credits s calculated based on our logical model. We assume that access will be denied for any values of $s_{card} \leq 0$, and according to the access control policy, an alert can be sent to security officers if the value of security credit drops lower than a certain level. Based on this approach, the multiplying effects of different irregular behaviors can be added up, and access will be denied if the resulting deduction causes a negative value of s_{card}. The concept of security credit can be used to derive a mathematical model that reflects the abnormal usage behaviors and establishes a quantitative relationship between the system alert levels. The following formula defines the amount of security credits to be deducted during access checks.

$$s_d(\{v_c\},\{v_b\},l) = c(\{v_c\}) \cdot b(\{v_b\}) \cdot r(l) \quad \text{Equation 2, where:}$$

s – Security credits, c – conditional function, b – behavior function, r – reduction factor function, $\{v_c\}$ – input vector of conditions factors,$\{v_b\}$ – input vector of behavior factors, l – security level

$c(\{v_c\})$ returns either a value of 0 or 1, depending on the violations being detected. If no violations occur, the conditional function returns a 0 value which will nullify the returning result of the function s.

The behavior function returns the corresponding alert level in a real number. An increasing positive return value corresponds to higher probability being an irregular action. One of the methods for obtaining the behavior function is based on empirical analysis of abnormal behaviors. The function $b(\{v_b\})$ is a generalized function which can be one of the following: a discrete function with one or more variables, a set of conditions that will return a mathematical output or a formula obtained from empirical or data-mining analysis. Similar to conditional function, input parameters of b, $\{vb\}$, are defined as a vector to generalize the input set. The reduction factor function $r(l)$ ensures the returning value of s falls within S. It introduces greater flexibility to the model by allowing security officers to define individual access control policies according to the required security level l.

Both conditional and behavior functions described in Equation 2 are dependent over several parameters.

A. Time variable

The current system time t_c, is an important parameter for real-time validation of access data. It is also recorded in the system and transmitted later to a centralized data server as an important source for data analysis.

B. Geographical variable

The location of the access points can be represented mathematically by the physical coordinates of the reference points. For real-time validation purposes, this parameter can be converted to a time-related factor in order to generalize the input values in Equation 2.

C. Access sequence

In physical access control systems, there might be a required access sequence. Consider again the example shown in Fig. 2. User should have permitted to enter the room through the doorways ap_{ix} before he can access the computer ac_{c1}.

D. Action Frequency

Some actions are not irregular when they are performed occasionally. However, if it repeats, its alert level must rise accordingly. For example, frequent visit to a room during a short period of time.

IV. EMPERICAL MODEL

We show some possible security checks formulae developed using Equation 2. Some generic input parameters are below:

Table 3. Generic input parameters

t_c: current time, t_l: last access time, t_o: start time of validation period, t_e: end time of validation period, s_{card}: security credits stored in card, l_{def}: defined security level, $ac(uz)$: function that returns the current access point of user uz, $al(uz)$: function that returns the last access point of user uz

When the card is presented to the relying party, a set of validations will be performed, and a new value of security credit stored in the card s_{card} will be calculated based on the following relationship:

$$s_{card} = s_{card} - s_d(\{v_c\},\{v_b\},l) \qquad \text{Equation 3}$$

If no irregularities are detected, function s_d will return a zero and the value of s_{card} will remain unchanged. Access permissions are controlled by the value of s_{card} via *permission validations* and *irregularities validations*.

Table 4. Permission validations checks

Card expiration check (Critical)

Description: Check whether card is expired

$c(\{v_c\}) = \phi\left(t_{card\_expiry} < t_c\right), \{v_c\} = \{t_{card\_expiry}, t_c\}$

$b(\{v_b\}) = \left|s_{card}\right|, \{v_c\} = \{s_{card}\}$

$r(l) = l_{def}, l = l_{def}$

t_{card_expiry}: expiry time of the card

Card suspension check (Critical)

Description: Check whether card is suspended

$c(\{v_c\}) = \phi\left(t_{spn\_time} \cdot NoRecSpn(uz) \le t_c\right), \{v_c\} = \{t_{spn\_time}, NoRecSpn, t_c\}$

$b(\{v_b\}) = \left|s_{card}\right|, \{v_c\} = \{s_{card}\}$

$r(l) = l_{def}, l = l_{def}$

t_{spn_time}: Reference time for suspension of the card

$NoRecSpn(uz)$: Function that returns the number of records matched in the suspended list

User permission level check (Critical)

Description: Check whether user has the required access permission level

$c(\{v_c\}) = \phi\left(lvl(uz) \neq lvl_{req}\right), \{v_c\} = \{lvl(uz), lvl_{req}\}$

$b(\{v_b\}) = \left|s_{card}\right|, \{v_c\} = \{s_{card}\}$

$r(l) = l_{def}, l = l_{def}$

$lvl(uz)$: Function that returns the level of a user uz

lvl_{req}: Required permission level

PIN Check (Critical)

Description: Check for correct PIN is inserted

$c(\{v_c\}) = \phi\left(PIN_{input} \neq PIN(uz)\right), \{v_c\} = \{PIN_{input}, PIN_{card}\}$

$b(\{v_b\}) = \left|s_{card}\right|, \{v_c\} = \{s_{card}\}$

$r(l) = l_{def}, l = l_{def}$

PIN_{input}: PIN inserted by the user

$PIN(uz)$: PIN value stored in the card of user uz

PIN trials check (Critical)

Description: Check for PIN input trials exceeds the possible number of PIN trials.

$c(\{v_c\}) = \phi\left(PIN_{trials}(uz) > PIN_{max\_trials}\right), \{v_c\} = \{PIN_{trials}, PIN_{max\_trials}\}$

$b(\{v_b\}) = \left|s_{card}\right|, \{v_c\} = \{s_{card}\}$

$r(l) = l_{def}, l = l_{def}$

$PIN_{trials}(uz)$: Current time of PIN trials of user uz

PIN_{max_trials}: Maximum possible times of PIN trials

Permission validations is considered as critical violations that cause immediate access denial if any of these checks fails. ***Irregularities validations*** perform checks of several parameters for detection of abnormal data patterns and will return the corresponding reduction factor calculated using Equation 2. If the cumulative value of s_{card} drops below a threshold defined by the security officer, an alert will be sent.

Table 5. Temporal violations checks

Check time period constrain (critical)

Description: Check if card is accessed during authorized time range $[t_o..t_e]$

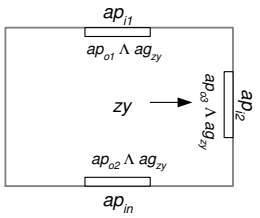
$c(\{v_c\}) = \phi\left(\left(t_c < t_o\right) \vee \left(t_c > t_e\right)\right), \{v_c\} = \{t_c, t_o, t_c\}$

$b(\{v_b\}) = \left|s_{card}\right|, \{v_c\} = \{s_{card}\}$

$r(l) = l_{def}, l = l_{def}$

Check zone overstay

Description: Assume $\{ap_{ik}\}_n$ as set of entrances of constrained one zy, and $\{ap_{oj}\}_n$ as set of exits. Check whether duration of stay in zy exceeds $t_{max}(z_y)$



$c(\{v_c\}) = \phi\left(al(uz) \subset \{ap_{ik}\}_n\right) \cdot \phi\left(ac(uz) \subset \{ap_{oj}\}_n\right) \cdot \phi\left(\left(t_c - t_l\right) \ge t_{max}(zy)\right),$
$\{v_c\} = \{ac(uz), al(uz), \{ap_{ik}\}_n, \{ap_{oj}\}_n, t_c, t_l, t_{max}(z_y)\}$

$b(\{v_b\}) = t_{max}(z_y) - \left(t_c - t_l\right), \{v_b\} = \{t_c, t_l, t_{max}(z_y)\}$

$r(l) = l_{def}, l = l_{def}$

$\{ap_{ik}\}_n$, $\{ap_{oj}\}_n$: set of entrances and exits of zone z_y

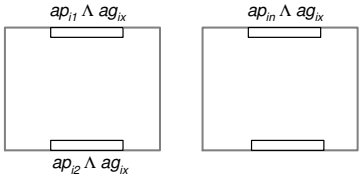
$t_{max}(z_y)$: estimated maximum time of stay in zone z_y

Repetitive trials violation is the abnormality that a user presents his card over the same relying party locating in nearby zones geographically repetitively over a short period of time. ***Displacement violations*** verify abnormal usage patterns between two or more access points over a period of time.

Table 6. Repetitive trials violations checks

Check irregular repetitive access

Description: Check whether the set of access points $\{ap_{ik}\}_n$ belonging to the group ag_{ix} is repetitively activated between the time period t_o and t_e



$c(\{v_c\}) = \phi\left(ac(uz) \in ag_{ix}\right) \cdot \phi\left(al(uz) \in ag_{ix}\right) \cdot \left[\left(t_c - t_l\right) > \left(t_e - t_o\right)\right],$
$\{v_c\} = \{ac(uz), al(uz), ag_{ix}, t_e, t_o, t_c, t_l\}$

$b(\{v_b\}) = \left[\left(t_e - t_o\right) - \left(t_c - t_l\right)\right] \cdot NumAc(ID(uz), ag_{ix}, t_o, t_e) \cdot Status(uz),$
$\{v_b\} = \{ID(uz), ag_{ix}, t_e, t_o, t_c, t_l\}$

$r(l) = l_{def}, l = l_{def}$

$ID(uz)$: Unique identification number of the user

ag_{ix}: access group to be verified

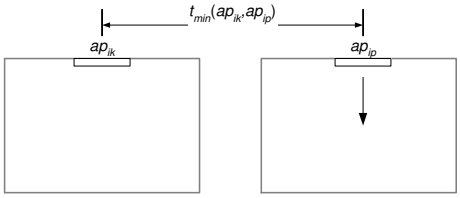
$Status(uz)$: User status function

$NumAc(ID(uz), ag_{ix}, t_o, t_e)$: Function that returns the number of access of a user on the access points belonging to ag_{ix} within the time period $[t_o..t_e]$

Table 7. Displacement violations checks

Check minimum access time between two points

Description: Check whether the access time of two access points ap_{ik} and ap_{jp}, is shorter than average minimum time required from moving between points ap_{ik} and ap_{jp}.



$c(\{v_c\}) = \phi\left(ta(ap_{ip}) - ta(ap_{ik}) \le t_{\min}(ap_{ip},ap_{ik})\right), \{v_c\} = \{ap_{ip},ap_{ik}\}$

$b(\{v_b\}) = t_{\min}(ap_{ip},ap_{ik}) - \left(ta(ap_{ip}) - ta(ap_{ik})\right), \{v_b\} = \{ap_{ip},ap_{ik}\}$

$r(l) = l_{def}, l = l_{def}$

$t_{min}(ap_{ip},ap_{ik})$: Function that returns average minimum time required from going between points ap_{ik} and ap_{ip}

$ta(ap_{ix})$: Function that returns the access time of point ap_{ix}

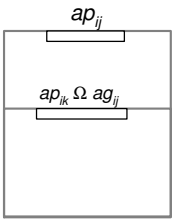
Sequential violations: Consider this example: If an user does not have permission to access ap_{ij}, unless using illegal methods or sneak in through doorway ap_{ij} by following another user with legitimate access, logically it is not possible to access ap_{ik} Λ ap_{ij} before ap_{ij} is activated. The formulae shown above are obtained by pure empirical analysis of some possible abnormal or irregular activities.

Irregular access patterns can be detected by using data mining techniques to build up rules that can capture the behavior. Some literatures have discussed about data mining for fraud and intrusion detection [1][2]. Two approaches can be used for irregular action detection analysis. Anomaly detection tries to find normal usage patterns from the audit data, while misuse detection is about finding and retrieving irregular access patterns using access information.

Table 8. Sequential violations checks

Out of sequence access (critical)

Description: Check whether ap_{ik} is accessed after ap_{ij} has been accessed.



$c(\{v_c\}) = \phi\left[ac(uz) = ap_{ik})\right] \cdot \phi\left(al(uz) \ne ap_{ij}\right),$

$\{v_c\} = \{ac(uz), al(uz), ap_{ik}, ap_{ij}\}$

$b(\{v_b\}) = \left|s_{card}\right|, \{v_c\} = \{s_{card}\}$

$r(l) = l_{def}, l = l_{def}$

ap_{ij}: access point ij

ap_{ik}: access point *ik* that must be activated after ap_{ij} has been accessed

When all access data are transferred to a centralized server, frauds or irregular access patterns can be extracted from this large pool of information by making use of data analysis and mining tools. Data mining generally refers to the process of extracting descriptive models from vast amount of data. The following types of algorithms are particular useful for mining our data to obtain useful data for refinement of our model.

Classification – categorize data into one of several predefined groups. These algorithms normally output "classifiers", for example, in form of decision trees or rules. By gathering sufficient "normal" and "irregular" access information, we can apply classification techniques to learn a classifier that can label or predict new unseen audit data as belonging to the normal class or the abnormal class. Using this information, we can formulae that reveal irregular access patterns according to Equation 2.

Sequence analysis – these algorithms model sequential patterns that try to discover what time-based sequences of access activities are frequently occurring together. These frequent event patterns provide guidelines for incorporation temporal statistical measure into our model.

V CONCLUSION

We developed a logical model that establishes a quantitative relationship of the level of irregular actions with the concept of security credits. This model is based on a physical access control environment where doors are equipped with smart-card devices. Here we assume that the model is based on a smart-card reading device attached to each door. In other words, the access points are oriented at the locations of the doors. A similar work was recently done in a physical RFID environment [4] where detection has a wider coverage as wireless sensors can flexibly installed around a closed compound. In this logical model, the security credit updates dynamically during the accesses. The corresponding number of credits could be interpreted as the degree of irregular behaviors. The centralized approach that gathers the relevant access information enables data analysis, that in turn can provides useful insights on the access patterns of users.

REFERENCES

[1] John F. Sowa, "Knowledge Representation, Logical, Philosophical and Computational Foundations," Brooks/Cole, Thomson Learning

[2] Wenke Lee, Salvatore J. Stolfo, Kui W. Mok, "A Data Mining Framework for Building Intrusion Detection Models", *IEEE Symposium on Security and Privacy*, Oakland, Califonia, May 9-12, 1999

[3] Bruce Schneier, Applied Cryptography, Second Edition – Protocols, Algorithms, and Source Code in C, John Wiley & Sons, Inc.

[4] Pengfan Yan, Robert P. Biuk-Aghai, Simon Fong, Yain-Whar Si, "Detection of Suspicious Patterns in Secure Physical Environments", *IEEE International Conference in Information Technology and Applications (ICITA2007)*, January 15-18, 2007, Harbin, China