Protecting on-line casinos against fraudulent player drop-out*

Joaquín García[1], Frédéric Cuppens[2], Fabien Autrel[3], Jordi Castellà-Roca[4],
Joan Borrell[1], Guillermo Navarro[1], and Jose A. Ortega-Ruiz[1]

[1] UCCD-UAB, 08193 Bellaterra (Catalonia), Spain

[2] GET-ENST-Bretagne, 35576 Cesson Sévigné, France

[3]ONERA-CERT, 31055, Toulouse Cedex, France

[4] ETSE-URV, 43007 Tarragona (Catalonia), Spain

Email: joaquin.garcia@uab.es, frederic.cuppens@enst-bretagne.fr,
fautrel@cert.fr, jcaste@etse.urv.es, {jborrell,gnavarro,jao}@ccd.uab.es

Abstract

Some multiplayer, on-line games rely on the collaboration of all participating players. If a player's gamble is aborted, the rest of players cannot continue playing. This behavior can be used by fraudulent players to avoid paying by simply quitting the game before its completion. It is difficult to decide whether a player has left the game in a deliberated, fraudulent way, because there are many factors, both intentional and inadvertent, that can cause the abandonment. This paper presents a fraud detection system that specially fits to such scenarios. By gathering and correlating information held by multiple sources, our approach will help the on-line casino administrator to decide if a player leaving a game is actually cheating. Results of our work can be easily adapted for use against other existing on-line gambling frauds.

1 Introduction

We shall use the term "on-line gambling" to refer to gambling services accessed via a remote connection, including the Internet, interactive television and mobile phones [8]. On-line gambling has attracted a high number of players and revenues during the last years, probably due to its space and time independence, which result in highly improved accessibility. Thus, in 2001, more than one million Americans per day logged on to the Internet and played casino-style games, or made sports wagers for real money [15]. The same year, the on-line gambling industry was valued at approximately two billion dollars and is projected to exceed six billion dollars by 2004 [1].

Despite its popularity, remote gambling is plagued by several open issues to be addressed to guarantee players and casino protection [8]. The technical and organizational e-commerce issues of government legislation relating to Internet gambling, transaction security and internal control procedures were briefly addressed in [11].

An outstanding problem is game aborting, or player drop-out. In most on-line games if one player drops out the rest of players cannot continue and the game must be aborted.

The *Interactive Gambling (Player Protection) Act* [14] tries to protect the players. Thus, if after making a wager in an authorized game conducted by a licensed provider, a player's participation in the game is interrupted by a technical failure that prevents the player from continuing with the game, the licensed provider must refund the amount of the wager to the player as soon as practicable. However, players can abuse this protection. For instance, a malicious player with a bad game hand can decide to make a denial of service attack against another player to avoid losing his money. The victim cannot continue playing and the game must be interrupted. As a result, the on-line casino must refund the amount of the wager to all the players and the cheater avoids paying.

In [12, 13] a fraud detection in an IP-based media-on-demand service is discussed. It is suggested that intrusion detection systems (IDS) can be used to detect fraud attempts

*Part of this work has been founded by the Spanish Government, through its projects TIC2003-02041 and SEG-2004-04352-C04-01, and the Catalan Government Department DURSI, with the grant 2003FI-126.

in a gambling site. Based on the same idea, we propose in this paper the utilization of a detection and reaction mechanism to detect if the player drop-out is due to a fraudulent attack. This information will help the on-line casino administrator to decide if a player drop-out is fraudulent.

The rest of this paper is organized as follows. Section 2 presents our detection mechanism based on alert correlation. This detection mechanism uses LAMBDA, a language suggested in [6] to model actions and objectives, and to detect incoming scenarios. In Section 3, we present a second mechanism to react on detected scenarios. Section 4 shows the application of a research prototype that implements these two mechanisms to prevent a fraudulent player drop-out scenario. Finally, the last section lists our conclusions.

2 Detection Mechanism

We present in this section a detection mechanism based on alert correlation. Correlating information held by multiple sources is a technique that has been discussed in several papers. Nevertheless, the goal aimed by those approaches are different and need to be explained.

With the rise of cooperative or distributed detection tools, the problem of reasoning on information coming from multiple sources spread across a monitored system is key. Correlating this information allows to fulfill different goals:

- *Information consolidation*: using a set of distributed detection probes increases the detection power, but usually provokes that each detected event raises multiple alerts. These alerts must be aggregated and fused, associating them to a unique source, before further processing.

- *Scenario detection*: the granularity of detection alerts is low. The event associated to an alert can be non malicious when considered alone but when a more global vision is adopted we may conclude that this elementary event is part of an ongoing scenario.

The notion of alert correlation as the process of aggregating alerts related to the same event has been studied in [7] and [3]. In order to aggregate alerts, they define a similarity relationship between alert attributes. A second approach to perform alert correlation, discussed in [4] and [2], is the process of detecting scenarios of alerts. In our proposal we use the latter approach, introducing the notion of alert correlation as the process of finding a set of alerts in the stream of detection alerts organized into a scenario. Our formalism is explained below.

2.1 Modeling Actions and Objectives

We can model the actions of a fraudulent player during an on-line game as the planning activity of an agent [4]. This agent's actions will be registered as facts about its usage of the on-line services. Often, these actions will follow a pattern indicating their illicit goals. Thus, we can use these indicators, in combination with other environment events, to predict the agent's fraudulent objectives, considered as a reachable set of objectives for the agent's plans. These goals constitute, of course, a violation of the system's security policy, which is accomplished via a concrete activity scenario. Our aim will be detecting such attack scenarios via associated sets of alerts detectable via a monitoring system.

We propose the use of the LAMBDA language [6] to model the possible agent actions leading to one or more (fraudulent) objectives. LAMBDA is the acronym for LAnguage to Model a dataBase for Detection of Attacks. This language provides a logical, synthetic description of generic actions, which are defined in terms of the following attributes:

- *Pre-condition*: defines the state of the system needed in to carry out the desired action.

- *Post-condition*: defines the state of the system after execution of the action.

- *Detection*: is a description of an alert expected whenever the action is executed.

- *Verification*: specifies the conditions to verify the success of an action.

The alerts launched by a detection system provide evidence of the occurrence of some events but are not sufficient to conclude that these events will actually cause a change in the state. This is why a LAMBDA description also includes a verification attribute that provides conditions to be checked to conclude that the execution of the action has been successful.

On the other hand, we define an *objective* as a specific system state characteristic of a violation of the security policy[4]. A LAMBDA description of an objective consists of just one attribute, namely:

- *State*: defines the state of the system that corresponds to a security policy violation.

2.2 Correlating Actions and Objectives

Reference [5] shows how to correlate detected actions to identify activity scenarios using LAMBDA. It is then possible to extrapolate such scenarios to predict future actions

Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05)
0-7695-2315-3/05 $ 20.00 **IEEE**

and even the objective pursued by a given set of actions. An extrapolation can yield several possible scenarios, which can be ordered from most to least likeness according to the algorithm described in [2].

The ordering algorithm uses correlations based on a unification principle on predicates [9]. We mainly rely on the concept of *direct correlation*.

Direct correlation represents the idea of positive influence between two actions. We say that an action a has a positive influence over an action b if a is directly correlated to b. In such a case, the effects of a, namely the set of predicates in $post_a$, allows to satisfy a subset of the pre-requisites of pre_b. More concretely, the concept of *direct correlation* is defined as follows. Let a and b be two LAMBDA descriptions of actions, and $post_a$ and $post_b$ the set of literals of *post-condition* in conjunctive form for, respectively, a and b.

Direct correlation: a and b are directly correlated if $\exists$ E_a and E_b such that:

- $(E_a \in post_a \wedge E_b \in pre_b)$ or $(not(E_a) \in post_a \wedge not(E_b) \in pre_b)$
- E_a and E_b are unifiable through a most global unifier θ.

It is also useful for our proposal to integrate in the detection process the preliminary steps that an agent can perform to collect data on a specific system, by means of the concept of *Knowledge gathering correlation* [5]:

Knowledge gathering correlation: a and b are knowledge gathering correlated if $\exists$ E_a and E_b such that:

- $(knows(Agent, E_a) \in post_a \wedge E_b \in pre_b)$ or $(knows(Agent, not(E_a)) \in post_a \wedge not(E_b) \in pre_b)$
- E_a and E_b are unifiable through a most global unifier θ.

The notion of *correlation unifier* allows us to apply on-line correlation. Since two actions a and b are correlated as soon as they have one predicate in common in $post_a$ and pre_b, we may have several unifiers for two actions. Thus, the set of unifiers allows us to know which action can be correlated with a given action under some unification condition between their variables. Applying on-line correlation consists in exploring the set of correlation unifiers each time a new alert is received, given that the alert corresponds to an instance of an action model. More concretely, we define the concept of *correlation unifier* as follows:

Correlation unifier: denoted Ξ_{ab}, is defined as the set of all possible unifiers of direct correlation or knowledge gathering correlation that correlate $post_a$ and pre_b.

Similarly, we can also apply the notion of correlation between an action and an objective. In this case, we can detect that an action may allow to reach or help to reach the fraud objective. We simply have to correlate the *post-condition* of an action and the *state condition* of an objective.

2.3 Detecting Scenarios

Once all the actions and objectives available for the agent have been modeled in LAMBDA, we can generate the set of unifiers between each pair of actions, and respectively between an action and an objective. This generation is done off-line. When an alert is received, we will bind this alert to an action model and then check for a unifier between the new and past alerts. If some unifier in the unifier set is identified, we can then say that the associated actions are correlated in the same scenario.

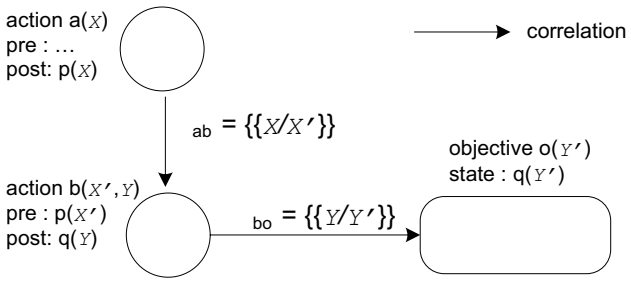

Figure 1. Sample correlation graph

Using this approach, it is possible to build a correlation graph, as shown in Figure 1, where nodes are LAMBDA descriptions and edges are correlation unifiers. In this example, we assume that two actions a and b are detected: an occurrence of a with argument $X = x$ and an occurrence of b with argument $Y = y$. The correlation process diagnoses that action a is correlated with action b, that action b is correlated with a given fraud objective o, and that this objective is achieved.

3 Reaction Mechanism

Just detecting activity scenarios does not prevent the agent from reaching its objective: we need an additional mechanism to decide when to execute a counter measure once the scenario has been partially observed. Through this mechanism, the next expected action will be blocked via an anti-correlated action. Using the formalism presented in [2]

Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05)
0-7695-2315-3/05 $ 20.00 **IEEE**

to model counter measures as actions launched by the system, counter measures will have attributes similar to those of actions. The main difference is that the *detection* attribute associated with a normal action is replaced by a new one, called *response*, which defines the corresponding attributes to perform the counter measure.

From the modeling point of view, the models for counter measures are not different from the ones representing the set of actions available to the agent. Actually, a counter measure is an action b anti-correlated with another action a, i.e., one of the predicates in a's post-condition is correlated with the negation of one predicate in the pre-condition of action b. More formally, using the notation of previous sections, we define anti-correlated actions as follows:

Anti-correlation: a and b are anti-correlated if $\exists$ E_a and E_b such that:

- $(E_a \in post_a \wedge not(E_b) \in pre_b)$ or $(not(E_a) \in post_a \wedge E_b \in pre_b)$
- E_a and E_b are unifiable through a most global unifier θ.

As for a correlation unifier, an anti-correlation unifier defines which actions can be anti-correlated, telling how the variables must be unified in the predicates involved in the anti-correlation link. More concretely, we define the concept of *anti-correlation unifier* as follows:

Anti-correlation unifier: denoted Ψ_{ab}, is the set of all possible unifiers θ that anti-correlate $post_a$ and pre_b.

Using the same approach, it is possible to define anti-correlation between a counter measure and an objective. We have just to replace *pre-condition* by *state* in the previous definition.

3.1 Reacting on Detected Scenarios

The anti-correlation mechanism relies on the use of the hypothesis generation mechanism [2]. Each time a new alert is received, the correlation mechanism finds a set of action models that can be correlated in order to form a scenario leading to an objective. This set of hypothesis is then instantiated into a set of *virtual alerts*.

When a scenario is identified, the correlation mechanism then looks for action models that can be anti-correlated with the virtual actions, and provides a graph of actions, virtual actions and objective. The set of anti-correlated actions becomes the set of counter measures available for the hypothesis represented by the partially observed scenario. Thus,

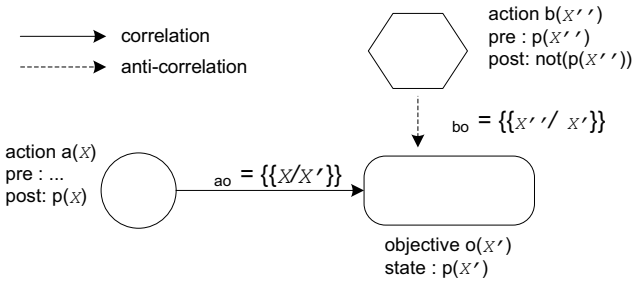

Figure 2. Sample correlation graph with reaction on objective

there is two possible prevention mechanisms. One that directly applies to virtual actions, and other that applies to the objective.

Figure 2 presents a sample correlation graph with direct reaction on objective. In this example, we assume that an action a, with argument $X = x$, is correlated with a given fraud objective o and this objective is achieved. The anti-correlation mechanism finds a possible counter measure candidate: an action b with parameter $X'' = X'$ to invalidate the condition p on the objective o. In our approach, this action is then suggested to the administrator as a possible counter measure to prevent the scenario's objective.

In the following section we show how to use the detection and reaction mechanism presented in Section 2 and Section 3 to design a prevention mechanism that detects and reacts to the fraudulent player drop-out scenario described in Section 1.

4 Preventing the Fraudulent Scenario

This section presents the Prevention Cells framework [10], a detection and reaction platform which implements the fraud prevention mechanism described in this paper. This framework is a research prototype implemented in C and C++.

Our platform works as follows. A set of gathering tools (*sensors*) take information at application, host, and network layer in various monitored *targets*. This audited data is then filtered and pre-formated into alerts by analyzers. These alerts are sent to an alert database. An enhanced version of CRIM [5] analyzes these alerts using the detection and reaction formalism presented in this paper.

CRIM is a correlation manager which implements the needed functions to cluster different alerts that correspond to the same occurrence of an action, as well as the functions to correlate and anti-correlate actions and objectives. Thus, to detect different scenarios, CRIM uses the approach based on correlation defined in Section 2. Likewise, CRIM uses

Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05)
0-7695-2315-3/05 $ 20.00 **IEEE**

Action $port\text{-}scan(A, H, P)$
Pre: $open(H, P)$
Detection: $classification(Alert,'TCP\text{-}Scan')$,
$source(Alert, Agent, A)$,
$target(Alert, Host, H)$,
$target\text{-}service\text{-}port(Alert, Port, P)$
Post: $knows(A, open\text{-}service(H, P))$
Verification: $true$

Action $winnuke(A, H, S)$
Pre: $use\text{-}os(H, windows)$,
$use\text{-}service(H,'Netbios')$,
$open(H, 139)$
Detection: $classification(Alert,'Winnuke')$,
$source(Alert, Agent, A)$,
$target(Alert, Host, H)$
Post: $deny\text{-}of\text{-}service(A, H)$
Verification: $unreachable(H)$

Action $player\text{-}drop\text{-}out(P, H, G, t)$
Pre: $playing\text{-}game(P, G)$,
$playing\text{-}from(P, H)$,
$not\text{-}activity(H, t)$
Detection: $classification(Alert,'Player\text{-}Drop\text{-}Out')$,
$additional\text{-}data(Alert, player, P)$,
$additional\text{-}data(Alert, host, H)$,
$additional\text{-}data(Alert, game, G)$,
$additional\text{-}data(Alert, timeout, t)$,
Post: $cancelled\text{-}game\text{-}by\text{-}drop\text{-}out(P, H, G)$
Verification: $unreachable(H)$

Action $cancelled\text{-}game\text{-}by\text{-}fraud(A, P, H, G)$
Pre: $cancelled\text{-}game\text{-}by\text{-}drop\text{-}out(P, H, G)$,
$deny\text{-}of\text{-}service(A, H)$
Response: $classification(Alert,'Player\text{-}Drop\text{-}Out\text{-}Fraud')$
$additional\text{-}data(Alert, agent, A)$,
$additional\text{-}data(Alert, player, P)$,
$additional\text{-}data(Alert, host, H)$,
$additional\text{-}data(Alert, game, G)$,
Post: $not(cancelled\text{-}game\text{-}by\text{-}drop\text{-}out(P, H, G))$
Verification: $true$

Objective $fraudulent\text{-}drop\text{-}out(A, P, H, G)$
State: $cancelled\text{-}game\text{-}by\text{-}drop\text{-}out(P, H, G)$
$deny\text{-}of\text{-}service(A, H)$

Figure 3. Player drop-out scenario

the approach described in Section 3 to generate the corresponding counter measures.

These counter measures are transmitted to a counter measure manager component to perform a post-processing of the received alerts before sending the corresponding reaction to a set of response units. These responses do not need to be necessarily automatic reaction functions. The counter measure manager could provide the administrator in charge with a report, so that the latter could decide the needed response (be it passive or active).

The set of targets for our proposed player drop-out scenario is made up by the expected actors involved in the game. To simplify the scenario, let us suppose just the following ones. The *Player's Software*, which provides to end users (players) the functionality required to take part in the game. Different information from its environment (CPU

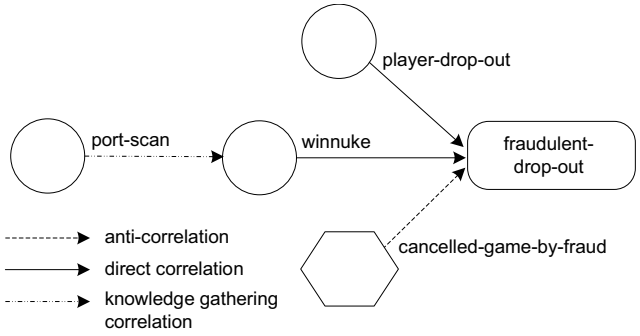

Figure 4. Correlation graph for the player drop-out scenario

and memory usage, operating-system, etc.) will be monitored. The *Internet and Network Service Providers*, which offer access to the on-line casino and bear the main responsibility for the game service. Since they serve as mediators between the player and the on-line casino, this target can offer technical data for the fraud detection process. Lastly, the *On-line Casino*. This target is an important actor in the prevention process since it delivers and receives the content that conforms the game.

To make the prevention system effective, the different components of our framework are distributed between the set of targets by using a publish-subscribe model [10]. Thus, the components of our framework are brokers and clients that are connected to them. The brokers themselves form the infrastructure used for the routing of notifications. Clients can publish notifications and subscribe to filters that are matched against the notifications traversing the broker network. When a broker receives a new notification, it checks if there is a local client that has subscribed to a filter matched by the notification. If so, the message is delivered to this client.

Let us present a detailed sample scenario for the player drop-out fraud. Corresponding to this scenario, Figure 3 shows a set of actions modeled using LAMBDA, together with the fraud objective. First, a fraudulent Agent *A* performs a port scanning of Host *H*. If port 139 is open, *A* concludes that the Operating System running on *H* is Windows and uses its NetBios service. Then, *A* can execute a Winnuke denial of service attack to *H*. This denial of service on *H* causes the Player *P*'s (which is playing in Game *G* from *H*) abandonment, when a timeout *t* is sent by the on-line casino. Actions $port\text{-}scan(A, H, P)$, $winnuke(A, H, S)$, and $player\text{-}drop\text{-}out(P, H, G, t)$ respectively give the description in LAMBDA of the port scanning, the denial of service, and the player drop out. The correlation of these actions, together with the objective

Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05)
0-7695-2315-3/05 $ 20.00 **IEEE**

$fraudulent\text{-}drop\text{-}out(A, P, H, G)$, conducts to the violation of the security policy (cf. Figure 4). On the other hand, the counter measure represented by the action $cancelled\text{-}game\text{-}by\text{-}fraud(A, P, H, G)$ is anti-correlated with the objective (cf. Figure 4). It notifies the administrator in charge that the real cause of *P*'s quitting is may be a fraudulent player drop-out instead a normal abandonment, and provides the corresponding attributes to perform the response.

5 Conclusions

We have presented in this paper a detection and reaction mechanism to identify if an on-line casino player is a victim of an attack. Our proposal uses an alert correlation and anti-correlation mechanism by exchanging information between the player's software, the on-line casino, and the different providers which collaborate to make possible the game. The correlation of this exchanged information is used to decide if a malicious player is trying to fraud the on-line casino.

Our approach is based on a logical representation of both fraud actions and counter measures. This logical representation has been integrated on an experimental prevention framework that collects and analyzes the alerts exchanged between the different peers.

Up to now, we only use our response mechanism to provide support to the administrator of the on-line casino. He takes the final decision to choose and launch a given response. This is a prudent strategy, but it introduces an overhead that is sometimes incompatible with real time response. Therefore, we are currently analyzing situations where it would be possible to *automatically* decide to launch the response.

References

[1] J. Baumgarten, A. Farr, and S. Brinkerhoff Proskauer Rose. Congress again confronts Internet gambling, cyberspace law. vol. 6 no. 6, September, 13 2001.

[2] S. Benferhat, F. Autrel, and F. Cuppens. Enhanced correlation in an intrusion detection process. In *Mathematical Methods, Models and Architecture for Computer Network Security*, Russia, September 2003.

[3] F. Cuppens. Managing Alerts in a Multi-Intrusion Detection Environment. In *17th Annual Computer Security Applications Conference New-Orleans*, New-Orleans, USA, December 2001.

[4] F. Cuppens, F. Autrel, A. Miège, and S. Benferhat. Recognizing malicious intention in an intrusion detection process. In *Second International Conference on Hybrid Intelligent Systems*, pp. 806–817, Santiago, Chile, October 2002.

[5] F. Cuppens, A. Miège, Alert correlation in a cooperative intrusion detection framework, In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pp. 202-215, Oakland, USA, May 2002.

[6] F. Cuppens and R. Ortalo. LAMBDA: A language to model a database for detection of attacks. In *Third International Workshop on the Recent Advances in Intrusion Detection (RAID'2000)*, Toulouse, France, 2000.

[7] H. Debar and A. Wespi. Aggregation and Correlation of Intrusion-Detection Alerts. In *Fourth International Workshop on the Recent Advances in Intrusion Detection (RAID'2001)*, Davis, USA, October 2001.

[8] Department for Culture Media and Sport of Great Britain. Gambling review body. http://www.culture.gov.uk/global/publications/archive_2001/gamb_rev_report.htm, July 17 2001.

[9] H.C.M. De Swart. *Mathematics, Language, Computer Science and Philosophy*, volume 2, Peter Lang, 1994.

[10] J. García, F. Autrel, J. Borrell, S. Castillo, F. Cuppens, G. Navarro, Decentralized publish-subscribe system to prevent coordinated attacks via alert correlation, In *Proceedings of the Sixth International Conference on Information and Communications Security*, pp. 223-235, Málaga, Spain, October 2004.

[11] M. Gregory, S. Michener, and P. Swatman. Internet gambling. In *"AusWeb'99" - the Fifth Australian World Wide Web Conference*, Ballina, NSW, April 17-21 1999. http://ausweb.scu.edu.au/aw99/ papers/gregory1.

[12] H. Kvarnstrm, E. Lundin, and E. Jonsson. Combining fraud and intrusion detection - meeting new requirements. In *In Proceedings of the fifth Nordic Workshop on Secure IT systems (NordSec2000)*, Reykjavik, Iceland, October 12-13 2000.

[13] E. Lundin. Aspects of employing fraud and intrusion detection systems. Master's thesis, School of Computer Science and Engineering, Department of Computer Engineering, Chalmers University of Technology, Gteborg, Sweden, 2002.

[14] Queensland Government. Interactive gambling (player protection) act. http://www.legislation.qld.gov.au /LEGISLTN/SLS/1998/98SL258.pdf, 1998.

[15] M. Schopper. Internet gambling, electronic cash and money laundering: The unintended consequences of a monetary control scheme. http://www.igcouncil.org/021007Schopper.pdf.

Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05)
0-7695-2315-3/05 $ 20.00 **IEEE**