An Intelligent Agent Security Intrusion System

J. PIKOULAS, W.BUCHANAN, *Napier University University, Edinburgh, UK.*
M.MANNION, *Glasgow Caledonian University, Glasgow, UK.*
K.TRIANTAFYLLOPOULOS, *Warick University, Warick, UK.*

ABSTRACT

Network security has now become one of the most important aspects in computer systems and the Internet. Apart from strong encryption, there is no definite method of truly securing network, thus they must be protected at different levels of the OSI model. At the physical layer they can be protected by lock-and-key, and at the data link they can be protected within VLANS (Virtual LANs). With the network and transport layers, networks can be secured by firewalls, which monitor source and destination network addresses, and source and destination ports, respectively. At the session level user names and passwords can be used. Unfortunately all these methods can be prone to methods which can overcome the protection used. This can be overcome by software which tries to detect the malicious moves of users, and try to inform the system administrators when this happen. Unfortunately there are too many factors involved in these systems. An important one is the human operator itself. Many security monitoring systems are also too complicated to run and maintain, and they generate big report lists that take weeks or months to be analysed. This paper expands the research previously undertaken on a misuse system based on intelligent agent software technology.

The system monitors user actions in real-time and take appropriate actions if necessary. Along with this our system used short-term prediction to predict the user behaviour and advise the system administrator accordingly, before the actual actions take place. This paper presents new results which are based on an increased number of users.

1 Introduction

At present computer security is a major worry for organizations, and this will continue as long as there is information, which can be stolen or damaged. The laws, which related to this type of crime are still being developed, and will take some time to implement. Many individuals think that the main problem relating to security is caused by external users (often known as hackers), but Carter and Catz [1] have shown that the primary threat come from individuals inside and organisation. Hence more emphasis should be placed on internal control mechanisms, such as audit log analysis.

In many cases, people that call themselves hackers [4] create security breaches. In the early days of computer hacking, these hackers did it for self-projection and not to improve their finances. Nowadays, there is a great deal of gain to be made from breaching system security, and these individuals tend to be:

1. **Professional programmers and IT specialists**. These individuals typically have extensive knowledge of the protocols and hardware that are used by organisations, work in teams, and have some inside knowledge of the organisational systems.
2. **Government agents**. These are typically ex-military using advanced information warfare methods (such as CIA software agents [3]).

However, no security measure guarantees a risk free environment, but increased security normally makes a system less easy-to-use. Many businesses must give access to parts of their system and make it easy to use, thus increasing potential exposure. Proper security controls require planning and careful implementation. Forestalling potential security breaches requires careful monitoring and management, and it is critical that these controls deter real problems. Unfortunately threats change as fast as both technology and business, thus adaptation and improvisation are key features of a security system.

No hardware or software element can ever be immune from security weaknesses. Many organisations will typically rely on a networking operating system for system security. To provide a measure of how secure a system is the US Government defines certain security levels: D, C1, C2, B1, B2, B3 and A1, which are published in the *Trusted Computer Security Evaluation Criteria* books (each of which has a different coloured cover to define their function). These include:

1. **Orange book**. Describes system security.
2. **Red book**. Interpretation of the Orange book in a network context.
3. **Blue book**. Application of Orange book to systems not covered in the original book.

Three techniques for illegal behaviour detection are commonly used in computer network security programs [5]: statistical anomaly detection, rule based detection, and hybrid detection, an amalgam of statistical anomaly detection and rule based detection. In most cases, the implementation of these techniques has been achieved by installing the security enhancement software on a centralized server. When this software crashes or is breeched, the complete network is at risk.

An alternative solution is to use agents to disperse network security management around the network. If the intrusion detection system is real time, it can detect the intrusion after the action, but never before. To address this problem we introduced a real time monitoring environment base on intelligent agent technology. For this we have constructed a software environment, that monitors the user behaviour and acts accordingly. We found that the real time monitoring had some limitations. The basic limitation to our initial system was that the system could detect a system anomaly, when the anomaly itself where started. That sometimes is not acceptable. So we started to research in ways to predict the actions of the user before they actually happen. For this a model was created with a new statistical model, which predicts the user future behaviour, or the system recourses. This statistical model is based on Bayesian statistics, and performs prediction on the same system resources that our agents are monitoring in real time.

The statistical model based on Bayesian multivariate regression proposed by Pikoulas and Triantafyllopoulos [8] takes user data behaviour and generates a predicted profile, so our intrusion system has sufficient information to foresee the future user actions.

2 Illegal Behaviour Detection Methods

2.1 Statistical Anomaly Detection

Statistical anomaly detection systems analyse audit-log data to detect abnormal behaviour. A profile of expected online behaviour for a normal user is predefined and derived from how an organisation expects a user to behave and from a system administrator's experience of the way a user is expected to use system resources. Typically, the audit logs are analysed and processed for statistical patterns of events for typical operations for which to determine usage patterns. These patterns are compared to the user's profile. With this, the administrator sets the expected user profile that is based on a model of how a user is expected to behave. The audit log is then analysed to look for statistical patterns of events to establish a typical operating pattern for a user; these are then compared with the user's profile.

The Safeguard project [6] adapted the NIDES statistical anomaly-detection subsystem to profile the behaviour of individual applications. Statistical measures were used to determine the proper usage of an application, and what differentiates this from inappropriate usage. A statistical score was assigned to the operation of applications and represented the degree to which current behaviour of the application corresponds to its established operational pattern. This system demonstrated the ability of statistical profiling tools and clearly differentiated the scope of execution among general-purpose applications. It showed that statistical analysis could be effective in analysing activities other than individual users, such as the system monitoring applications rather than users.

The system warns administrators that there has been a possible intrusion when a profile is different to a usage pattern. A major drawback with statistical anomaly detection is that this technique cannot predict extreme changes in user behaviours.

2.2 Rule Based Detection

Rule-based detection systems use a set of rules that define typical illegal user behaviour. These rules are formed by analyzing previous different patterns of attack, and analyses the audit-log data of a particular user and comparing the user's pattern with the rules. The drawback of this system is that the basic rules are predefined by system administrators, and cannot detect any new attack techniques. If a user exhibits behaviour that is not prescribed by the existing rules, the user can harm the system without being detected. The IDES system [7] is security enhancement software that stores knowledge about a system's known vulnerabilities, its security policies and information on previous intrusions. The information it uses to determine the network state is limited to the data packet header. As it does not examine the contents of the data packet, it may miss critical information about the nature of the data that goes throughout the network. It also scales very poorly where many machines are on a high-speed network.

Kumar and Spafford's model [9] uses pattern matching, as attacks can be classified as patterns, which match against occurrences (status of the system at that moment) in the system. These patterns can encode dependencies between system conditions and temporal conditions. Crosbie and Spafford's use autonomous agents [10], which are trained to detect anomalous activity in network system traffic. A drawback of this approach is that the system requires considerable training by a human operator before it becomes effective.

2.3 Hybrid Detection

Hybrid detection systems are a combination of statistical anomaly detection and rule-based detection systems. These, typically, use rules to detect known methods of intrusion and statistical based methods to detect new methods of intrusion.

CMDS (Computer Misuse Detection System) [11] is a security- monitoring package that provides a method to watch for intrusions. It detects and thwarts attempted logins, file modifications, Trojan horse installation, changes in administrative configurations and many other signs of intrusion. In addition, it constantly monitors for difficult detection problems like socially engineered passwords, trusted user file browsing and data theft that might indicate industrial espionage. CMDS supports a wide variety of operating systems and application programs. The drawback of this system is that it uses statistical analysis to make additional rules for the system. This is a drawback, as it can only detect attack patterns that have been used in the past and being identified as attack patterns, or predefined by the system operators. It also generates long reports and graphs of the system performance that requires to be interpreted by a security expert.

3 Forecasting methods

There are many forecasting methods for short to intermediate term analysis-forecasting [12] including multiple regression analysis, non-linear regression, trend analysis and decomposition analysis.

3.1 Bayes and empirical Bayes methods

Bayes and empirical Bayes (EB) [14] methods combine information from similar components of information and produce efficient inferences for both individual components and shared model characteristics. Many complex applied investigations are ideal settings for this type of synthesis. Recent advances in computing and the consequent ability to evaluate complex models have increased the popularity and applicability of Bayesian methods. Bayes and EB methods can be implemented using modern Markov chain Monte Carlo

(MCMC) computational methods. Properly structured Bayes and EB procedures typically have good frequentist and Bayesian performance, both in theory and in practice. This in turn motivates their use in advanced high-dimensional model settings (for example in longitudinal data or spatial-temporal mapping models), where a Bayesian model implemented via MCMC often provides the only feasible approach that incorporates all relevant model features.

4 Bayesian Intrusion Detection System

The proposed system is a hybrid intrusion detection system and was first described in Pikoulas, *et al* [15]. The system uses a hybrid detection technique. Invalid behaviour is determined by comparing a user's current behaviour with their typical behaviour and by comparing their current behaviour with a set of general rules governing valid behaviour formed by systems administrators. Typical behaviour is contained in a user historical profile. Prediction is computed using a Bayesian multivariate statistical model. The novelty of the approach is that the system uses a distributed architecture using intelligent software agent technology.

A user agent resides in a user workstation, and there is a core agent that resides on the system server. Each user agent has a variety of functions. When a user logs onto a workstation, the user agent retrieves the user login name and contacts the core agent. The user end agent gets the users profile from the core agent. After the user end agent retrieves the specified user profile starts to monitor the user behaviour.

The user profile file contains data describing the predicted specified user behaviour which the system administrators can create using the generic profiles of a group of users and adding rules drawn from individual behaviour patterns. Hence a user profile contains rules that describe the legal past behaviour of the user and the statistical predictions from these rules, up to the last time that the used logged on to the system. User behaviour is a collection of data from system variables including:

1. Information about software applications that have been used during a login session.
2. The path that these applications are running from.
3. The directory in which the user is currently working in.

The user agent monitors user behaviour until the user logs off. Ten seconds has been found to be a satisfactory interval because it gives an accurate description of user behaviour without compromising the performance of the system. Monitoring user behaviour is achieved using a C++ DLL within the user end agent. The DLL is called through a thread, which is set to run with the lowest priority, so the effect in the performance of the system will be at minimum. After the user end agent takes the user behaviour snapshot, it compares it with the historical profile (acquired from the core agent at the beginning of the user session). If any behaviour differences are found, it generates warning alerts to the user and also sends an alert signal to the core agent in order to inform the system administrators. If the system administrator accepts this change in the user behaviour, it is added to the user profile as a permitted rule, so the next time that the user tries to perform the same operation, no alerts will be generated. The rule is added instantly and not from the next time that the user is logged in to the system. When a user logs off, the user end agent sends a message to the core agent that this user is going to log off so it will update its user profile file with the prediction data, and store the new prediction data that generated for further process.

These data are then taken from our prediction model and processed, and stored to the system as prediction rules. These prediction rules are loaded from the user end agent, every time that the user logs to the system, so that the user end agent will be able to take appropriate actions if needed. We used a Bayesian multivariate statistical model because our problem is a linear multivariate problem and it is faster and more accurate to use a linear model than a non-linear model like neural networks [16].

5 Implementation

We built intelligent agent security enhancement software system, in which a core software agent resides on one server in a Windows NT network system and user end software agents reside in each user workstation. The software for each type of agent was written in SUN Java JDK Version 1.2 on a Microsoft Windows NT Version 4 environment running over a 10/100 Mbps network. There was one server and 10 clients. Figure 1 shows a core agent communicating with many user agents. A communication thread is a unique process that the core agent creates to transmit data to the user end agent in response to message transmitted from the user end agent. Unique processes enable the core agent to communicate with each user agent effectively and efficiently thereby enabling a fast response to network monitoring. Once the core agent has responded to a user agent, the process is killed. Figure 1 shows that a user agent implemented as four components:

1. **A sensor**. The sensor monitors the various software applications (for example, a word processor, a spreadsheet) that are currently being run by the user on that workstation. When a user logs in the sensor polls the user's activity every ten seconds and records the user's identifier and each application's name and process identifier. (The frequency of the sensor can be modified from the system administrator.)
2. **A transmitter**. After the first polling by the sensor, the transmitter sends this information to the core agent. The core agent then responds by sending a user historical profile. With an audit-log file for a period of one month, we observed that the size of an average user profile was between 400KB and 600KB, with a download time of between three and five seconds.
3. **A profile reader**. The profile-reader reads the user's historical profile.
4. **A comparator**. The comparator compares the user's historical profile with the information read by the sensor. If the current behaviour profile does not fall within the accepted behaviour pattern defined by the user historical profile, the comparator provides the transmitter with the following information that is then sent to the core agent: user identifier, invalid behaviour type and corresponding invalid behaviour type data. For example, if the invalid behaviour type were an *unauthorised directory access*

then the invalid behaviour type data would be the name of the directory attempting to be accessed. When invalid behaviour occurs, several courses of action are available, such as:

1. Warning message to the system administrator or end user.
2. Kill the specific application that has caused invalid behaviour.
3. Prevent the end user from running any further applications. In Case 1, the user agent informs the core agent and the core agent informs the systems administrator. The user agent terminates when a user logs off. Cases 2 and 3 can be achieved locally at the client workstation.

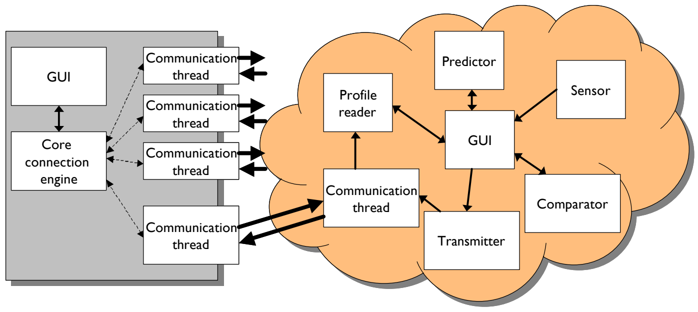

Figure 1 Agent Environment Topology

6 Experiments and Results

In our previous experiments, there was only one user with a limited observation of 10 times. This time the experiments are more generic. A previous paper [15] illustrated that the model worked. This has now been expanded to three users and with 100 observations. With this the prediction model makes 50 observations and adapt to the user behaviour, and then apply the prediction model to the next 50 observations.

In the results, a graph is constructed of the last 50 real observation values and the predictions, and in the y-axis plots time. Users used the typical applications, such as MS Word, MS Excel, MS Outlook, MS Internet Explorer and Borland C++. The users were free to use other applications and system recourses if they wanted to, but our system only monitored the above applications. There were no restrictions on when they use it, or how much time.

For these experiments, it is assumed that only the first application was legal to be used by all the users, and the rest were available to the system but restricted from the users.

As with previous experiments the time period of one hour is used to make the prediction. Thus, observation values for prediction are taken every one-hour, but real time monitoring system was still getting observations in real time.

We can observe the results of our experiments from the three graphs that follow. The first two graphs show the results without any sort of intervention. It can also be seen that our model is capturing all the different anomalies, with sometimes, luck of capturing the length of the high peaks of the real data graphs. Let's see first what the graphs represent. As we mentioned before we measured some user behaviour over a period of time. That is, when the user uses a specific system resource like using an application or using a system resource. That is what the graphs represent. In the 'x' axis we have the number of observations represented with a purple dotted line. This is the number of how many times we monitored the user to see what exactly he or she was doing in the monitored computer environment. And in the 'y' axis of the graphs we have the amount of time that the user is using the particular monitored recourse, for this observation, and it is represented with a dashed dark blue line. The value that the 'y' axis can take are between zero and one, since we explained that in this set of our experiments we are monitoring the users every one hour in order to see what they are using. This observation is different than the one that the agent is doing for the real time monitoring of the remote system. This timing of the user behaviour is every five seconds.

The graphs show prediction results that are taken from our agent environment. In Figure 2 and Figure 3 we can see that our model is able to follow the real user behaviour. In the graphs the dotted purple line is the real user behaviour and in dashed dark blue is our prediction. In Figure 2 we can see that even if our prediction (line in dashed dark blue) is not exactly matching the real user behaviour, our system could predict the spikes of the user behaviour, up to an extent. Examining Figure 2 we can see that the length of the spike is not very important to predict as to when it is going to occur. That is what proving that our model works. Although our prediction is accurate, as we evaluated Figure 2 above, we have introduced the notion of the intervention that we apply in our last result figure (Figure 4). The purpose of introducing intervention if we require more accuracy and more information on our prediction, like if we want to know exactly not only when it is going to occur, but even how long is going to take.

We can observe that the two trends are following each other. That means that our model is very accurately predicting future user behaviour.

In all of our examples we can see that our model is able to detect the deviations in the user behaviour, with maximum effects in the case of the use of intervention (see Figure 4).

In other words, the results of the experiments show that our model works. We can see that be following the two results lines on Figure 4. By how close they are, we can determine how good our model followed the real values, which are the real behaviour that the user followed. This is very important for our prediction system; because it can not only protect vital resources of the computer system, but also provide accurate help to the user, by knowing or predicting correct what are the next steps that the user will take.

7 Conclusions and Future Work

As we can observe from the graphs of the experiments, that our model applied on the observations of application five, with intervention, is very close to the actual user observations. We can observe the similarities of our results with the actual data. After applying our intervention mechanism, our prediction values are getting very close to the actual user behaviour (Figure 4). Short-term prediction is very difficult and there is large extend in research, trying to perfect it, especially financial institutions, weather prediction stations, stock exchange organizations, the army. Most prediction techniques are very inaccurate in short-term prediction or too slow or complicated to be used. Our proposed model can produce accurate results with a minimum amount of observations.

In future work are planning to perform more extensive experiments that will involve more users and larger number of applications. Our aim is to demonstrate that the model works in most cases, and also to refine the model.

There are aspects of the model that needs further research; especially in the way that intervention is applied, as this is not very practical. Also the values that are use to populate the values that are used in the intervention model are generic values that do not have the best results for all observed values.

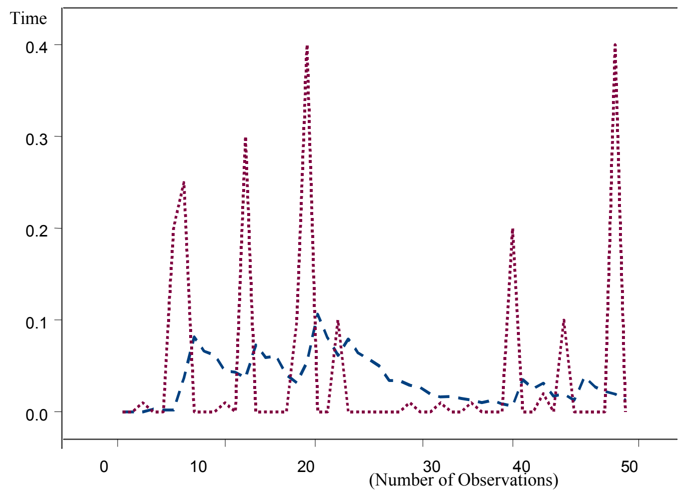

Figure 2 Application one Prediction and real values

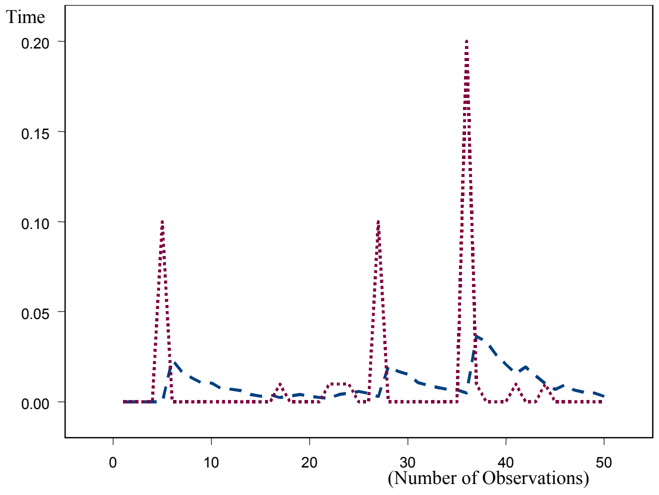

Figure 3 Application two Prediction and real values

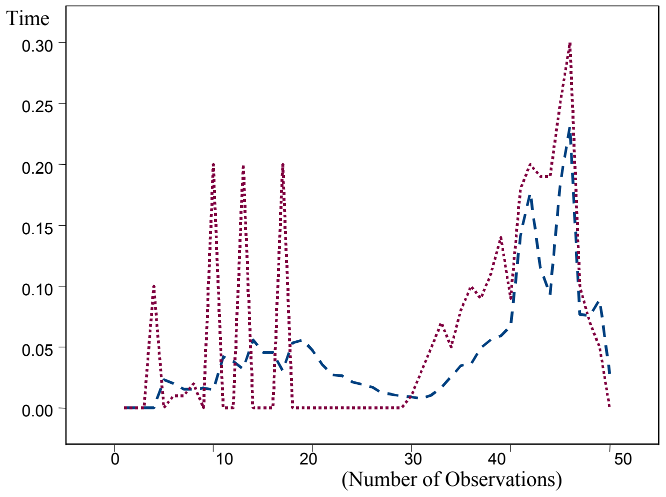

Figure 4 Application five Prediction and real values (with intervention)

8 References

[1] Carter and Catz, *Computer Crime: an emerging challenge for law enforcement*, FBI Law Enforcement Bulleting, pp 1-8, December 1996.

[2] Scotty Strunk, *Intrusion Detection FAQ*, SANS Institute Resources, December 2, 1999.

[3] Sunday Times, *CIA Spies On Europe*, August 4, 1996, London, UK.

[4] Roger Blake, *Hackers in The Mist,* Northwestern University, December 2, 1994.

[5] Chris Herringshaw, *Detecting Attacks on Networks*, IEEE Computer Magazine, pp 16–17, Dec. 1997.

[6] Debra Anderson, *Detecting Unusual Program Behavior Using the NIDES Statistical Component*, IDS Report SRI Project 2596, Contract Number 910097C (Trusted Information Systems) under F30602-91-C-0067 (Rome Labs), 1995.

[7] T. Lunt, H. Javitz, A. Valdes, *et al. A Real-Time Intrusion Detection Expert System (IDES)*, SRI Project 6784, Feb. 1992. SRI International Technical Report.

[8] J Pikoulas and K Triantafyllopoulos, *Bayesian Multivariate Regression for Predicting User Behaviour in a Software Agent Computer Security System*", 20th International Symposium on Forecasting, Lisbon, Portugal, June 21, 2000.

[9] Sandeep Kumar and Gene Spafford, *A Pattern Matching model for Misuse Intrusion Detection*, Proceedings of the 17th National Computer Security Conference, Oct. 1994.

[10] Mark Crosbie and Gene Spafford, *Active Defence of a Computer System using Autonomous Agents*, COAST Group, Dept. of Computer Science, Prudue University, Technical Report (95-008),2–3, Feb 1995.

[11] *The Computer Misuse Detection System*, http://www.cmds.net/, 1998.

[12] Professor Hossein Arsham, *Statistical Data Analysis: Prove it with Data*, University of Baltimore, http://ubmail.ubalt.edu /~harsham/statdata/opre330.htm

[13] The Bayesian Institute, *Bayes Thomas*, http://www.bayesian.org/bayesian/bayes.html, 7/5/2000.

[14] Carlin B. and T. Louis, *Bayes and Empirical Bayes Methods for Data Analysis*, Chapman and Hall, 1996.

[15] Pikoulas J, Mannion M and Buchanan W**, *Software Agents and Computer Network Security***, the 7th IEEE International Conference on the Engineering of Computer Based Systems, pp

211 – 217, Apr. 2000.

[16] Georges A. Darbellay and Marek Slama, *Forecasting the sort term demand for electricity. Do neural networks stand a better chance?*, International Journal of Forecasting, pp. 71-83, 2000.

[17] National Bureau of Standards, NBS FIPS PUB 46, *Data Encryption Standard*, National Bureau of Standards, U.S. Department of Commerce, Jan. 1977.

[18] M.J. Wiener, *Efficient DES Key Search*, TR-244, School of Computer Science, Carleton University, May 1994.

[19] M. Blaze, W. Diffie, R. Rivest, B. Schneier, T. Shimomura, E. Thompson, and M. Weiner, *Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security*, Jan 7 1996.

[20] National Institute of Standards and Technology, *Announcing Development of a Federal Information Standard for Advanced Encryption Standard*, Federal Register, v. 62, n. 1,2 Jan 1997, pp. 93-94.

[21] National Institute of Standards and Technology, *Announcing Request for Candidate Algorithm Nominations for the Advanced Encryption Standard (AES)*, Federal Register, Vol. 62, No. 117, Sep. 1997.

[22] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hal Niels Ferguson, *Two fish: A 128-Bit Block Cipher*, June 1998.