International Journal of Science and Engineering Research (IJ0SER),

Vol 6 Issue 4 April -2018
3221 5687, (P) 3221 568X

Seen: A Selective Encryption Method To Ensure Confidentiality For Big Sensing Data Streams

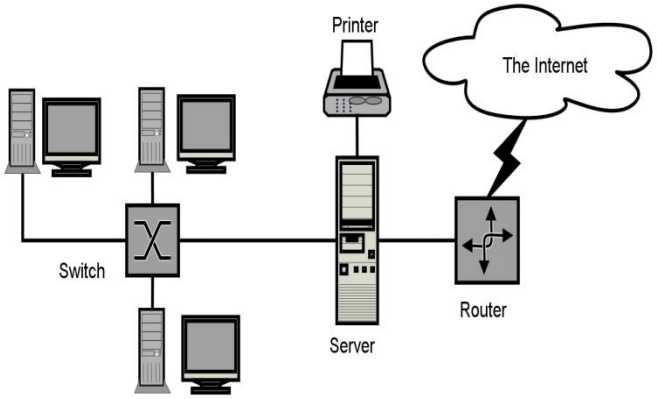
. [1] **T.Annapoorani,** [2]**J.Ashwini,** [3]**S.Suresh Kumar**
[1] [2] [3]Students SREC, [3]Assistant professor
Sri Ramakrishna Engineering College,
Tamilnadu,India

Abstract-Privacy has become a considerable issue when the applications of big data are dramatically growing in cloud computing. The benefits of the implementation for these emerging technologies have improved or changed service models and improve application performances in various perspectives. However, the remarkably growing volume of data sizes has also resulted in many challenges in practice. The execution time of the data encryption is one of the serious issues during the data processing and transmissions. To ensure the confidentiality of collected data, there is a need to prevent sensitive information from reaching the wrong people by ensuring that the right people are getting it. Sensed data are always associated with different sensitivity levels based on the sensitivity the sensed data types. In this paper, we propose a Selective Encryption (SEEN) method to secure big sensing data streams that satisfies the desired multiple levels of confidentiality. This proposed method is based on two key concepts: common shared keys and selective encryption based on the type of data. Theoretical analyses and experimental results of the SEEN method show that it can significantly improve the efficiency and buffer usage at DSM without compromising the confidentiality and integrity of the data streams. Additionally the proposed system provides time based access control among the data authenticators. So that data client can access the data at specified time period which is defined by the data owner. Hence it provides effective bandwidth management when compared to our existing system.

INTRODUCTION

What is the cloud? Where is the cloud? Are we in the cloud now? These are all questions you've probably heard or even asked yourself. The term "cloud computing" is everywhere.

In the simplest terms, cloud computing means storing and accessing data and programs over the Internet instead of your computer's hard drive. The cloud is just a metaphor for the Internet. It goes back to the days of flowcharts and presentations that would represent the gigantic server-farm infrastructure of the Internet as nothing but a puffy, white cumulus cloud, accepting connections and doling out information as it floats.



What cloud computing is not about is your hard drive. When you store data on or run programs from the hard drive, that's called local storage and computing. Everything you need is physically close to you, which means accessing your data is fast and easy, for that one computer, or others on the local network. Working off your hard drive is how the computer industry functioned for decades; some would argue it's still superior to cloud computing, for reasons I'll explain shortly.

The cloud is also not about having a dedicated network attached storage (NAS) hardware or server in

residence. Storing data on a home or office network does not count as utilizing the cloud. (However, some NAS will let you remotely access things over the Internet, and there's at least one brand from Western Digital named "My Cloud," just to keep things confusing.) For it to be considered "cloud computing," you need to access your data or your programs over the Internet, or at the very least, have that data synced with other information over the Web. In a big business, you may know all there is to know about what's on the other side of the connection; as an individual user, you may never have any idea what kind of massive data processing is happening on the other end. The end result is the same: with an online connection, cloud computing can be done anywhere, anytime

PROBLEM DEFINITION

Penetrations of big data techniques have further enriched the channels of gaining information from the large volume of mobile apps' data across various platforms, domains, and systems. Being one of technical mainstreams has enabled big data to be widely applied in multiple industrial domains as well as explored in recent researches. Despite many benefits of using mobile cloud computing, there are great concerns in protecting data owners' privacy during the communications on social networks. One of the privacy concerns is caused by unencrypted data transmissions due to the large volume of data considering an acceptable performance level, many applications abandon using cipher texts in mobile cloud data transmissions. This phenomenon can result in privacy leakage issues since plain texts are unchallenging for adversaries to capture information in a variety of ways, such as jamming, monitoring, and spoofing. This privacy issue is exigent because it faces to a contradiction between the security levels and performance that is usually attached to timing constraints

DRAWBACKS

- Communication overhead is high
- Lack of security
- Bandwidth consumption is high

PROPOSED SYSTEM

The proposed method is based on a typical shared key that is initialized and updated by a DSM. It maintains different levels of data confidentiality along with data integrity. The main contributions to develop and design a novel selective encryption with time based access control is to secure and maintain confidentiality of big sensing data streams in cloud according to different data sensitivity levels. It performs seamless refreshing of the shared key without disrupting ongoing data encryption or decryption. proposed model adopts different keys for the three levels of data confidentiality (i.e. no confidentiality, partial confidentiality and strong confidentiality) based on the data sensitivity levels. This model ensures the end-to-end security by protecting data from source device to cloud processing layer. Additionally It provides time based access control among the data authenticators. So that data client can access the data at specified time period which is defined by the data owner. Hence it provides effective bandwidth management when compared to our existing system.

ADVANTAGES

- High transmission rate because there is no need to encrypt un important data
- Less bandwidth consumption due to flexible encryption
- Data confidentiality
- It provides time based access control
- Flexible secured key generation

CONFIDENTIALITY

The main idea is that while encrypting the data packets at the source, we attach sensitivity level of data to each individual data packet. In the SEEN method, we apply different keys to encrypt the data packets for different data sensitivity levels. The aim is to provide different confidentiality levels based on the applications as well as the sensitivity levels of the data. in this project, we are considering three levels of data confidentiality: strong confidentiality, partial confidentiality, and no confidentiality; and two keys (i.e. k1, k2) for encryption methods.

DATA STREAM MANAGER

The DSM is responsible for performing the security verification of the incoming data streams in near real time to synchronize. In this architecture, the data streams are always in the encrypted format when they arrive at the DSM. The main idea is that while encrypting the data packets at the source, we attach sensitivity level of data to each individual data to perform the security verification at DSM by providing an end-to-end security of big sensing data streams. It is also important to perform a security verification of a data stream before the stream query processing in order to maintain the originality of the data

KEY GENERATION

Based on the data sensitivity or confidentiality level, the encrypted data stream is encrypted with the sensitivity levels. The strong encryption method uses k1 and is used to provide strong confidentiality, and the weak encryption method uses K2 to support partial confidentiality. Note that we do not need to encrypt the data packets for no confidentiality.

TIME BASED ACCESS CONTROL

Shared keys used for encrypting data packets. Users can only access the files that are all stored by the data owner within a predefined time period. Hence a user has to provide the access permission request to the authority. No one can access the data without the access permission and in expired time period. At the same time, effective bandwidth is always maintained.

INTRUSION DETECTION

The identity management is enforced to ensure that right level of access is only granted to the right person. The intrusion detection section is used to intimate the cloud management team, data centers and also its security pools about the intrusions by raising the alarms. The dangers which will happen by the intrusions are scalable .The rejection and warning messages/e-mails will be composed to send the client. The process starts with a possible intrusion event (this could be an unauthorized access) which triggers to compose email/message to the cloud data administrator immediately noted as the client process in this model.

LEDS: Providing Location-Aware End-to-End Data Security in Wireless Sensor Networks

Providing desirable data security, that is, confidentiality, authenticity, and availability, in wireless sensor networks (WSNs) is challenging, as a WSN usually consists of a large number of resource constraint sensor nodes that are generally deployed in unattended/hostile environments and, hence, are exposed to many types of severe insider attacks due to node compromise. Existing security designs mostly provide a hop-by-hop security paradigm and thus are vulnerable to such attacks. Furthermore, existing security designs are also vulnerable to many types of Denial of Service (DoS) attacks, such as report disruption attacks and selective forwarding attacks and thus put data availability at stake. In this paper, we seek to overcome these vulnerabilities for large-scale static WSNs. We come up with a location-aware end-to-end security framework in which secret keys are bound to geographic locations and each node stores a few keys based on its own location. This location-aware property effectively limits the impact of compromised nodes only to their vicinity without affecting end-to-end data security. The proposed multifunctional key management framework assures both node-to-sink and node-to-node authentication along the report forwarding routes. Moreover, the proposed data delivery approach guarantees efficient en-route bogus data filtering and is highly robust against DoS attacks. The evaluation demonstrates that the proposed design is highly resilient against an increasing number of compromised nodes and effective in energy savings.

WIRELESS attentions recently due to their broad applications in sensor networks (WSNs) have attracted a lot of both military and civilian operations. WSNs usually consist of a large number of ultra small low-cost battery-powered devices that have limited energy resources, computation, memory, and communication capacities and according to different applications such as battlefield reconnaissance and homeland security monitoring, WSNs are often deployed in a vast terrain to detect events of interest and deliver data reports over multi hop wireless paths to the sink. Data security is essential for these mission-critical applications to work in unattended and even hostile environments. One of the most severe security threats in WSNs is security compromise of sensor nodes due to their lack of tamper resistance.

In WSNs, the attacker could compromise multiple nodes to obtain their carried keying materials and control them and thus is able to intercept data transmitted through these nodes thereafter. As the number of compromised nodes grows, communication links between uncompromised nodes might also be compromised
through malicious cryptanalysis. Hence, this type of attack could lead to severe data confidentiality compromise in WSNs. Furthermore, the attacker may use compromised nodes to inject bogus data traffic in WSNs. In such attacks, compromised nodes pretend to have detected an event of interest within their vicinity or simply fabricate a bogus event report claiming a nonexistent event at an arbitrary location. Such insider attacks can severely damage network function and result in the failure of mission-critical
applications. Such attacks also induce network congestion and wireless contention and waste the scarce network resources such as energy and bandwidth, hence, severely affecting both data authenticity and availability. Lastly, the attacker could also use compromised nodes to launch a selective forwarding attack , in which case compromised

nodes selectively drop the going-through data traffic and, thus, data availability can be severely damaged. The existence of the aforementioned attacks together with the inherent constraints of sensor nodes make it rather challenging to provide satisfying data security in WSNs with respect to all its three aspects, that is, confidentiality, authenticity, and availability. In this paper, we propose an integrated security design providing comprehensive protection over data confidentiality, authenticity, and availability. Our design establishes a location-aware end-to-end data security (LEDS) framework in WSNs. The contributions of LEDS are outlined below. First, we propose a novel location-aware multifunctional
key management framework. In LEDS, the targeted terrain is virtually divided into multiple cells using the concept of a virtual geographic grid. LEDS then efficiently binds the location (cell) information of each sensor into all types of symmetric secret keys owned by that node. By this means, the impact of compromised nodes can be efficiently confined to their vicinity, which is a nice property absent in most existing security designs. What the attacker can do is to misbehave only at the locations of compromised nodes,
by which they will run a high risk of being detected by legitimate nodes if effective misbehavior detection mechanisms are implemented. Second, LEDS provides end-to-end security guarantee.

Every legitimate event report in LEDS is endorsed by multiple sensing nodes and is encrypted with a unique secret key shared between the event sensing nodes and the sink. Furthermore, the authenticity of the corresponding event sensing nodes can be individually verified by the sink. This novel setting successfully eliminates the possibility that the compromise of nodes other than the sensing nodes of an event report may result in a security compromise of that event report, which is usually the case in existing security designs.
Third, LEDS possesses an efficient en-route false data filtering capability to deal with the infamous bogus data injection attack. As long as there are no more than t compromised nodes in each single area of interest, LEDS guarantees that a bogus data report from that cell can be filtered by legitimate intermediate nodes or the sink deterministically. Effective en-route filtering of bogus data packets also results in significant energy savings as unnecessary forwarding is eliminated. Last, LEDS provides high-level assurance on data availability by dealing with both report disruption attack and selective forwarding attack simultaneously. By taking advantage of the broadcast nature of wireless links, LEDS adopts a one-to-many data forwarding approach, which is fully compatible with the proposed security framework.

Through exploiting the static and location aware nature of WSNs, we came up with a location-aware end-to-end security framework to address the vulnerabilities in existing security designs. In our design, the secret keys are
bound to geographic locations, and each node stores a few keys based on its own location. This location-aware property successfully limits the impact of compromised nodes only to
their vicinity without affecting end-to-end data security. Furthermore, the proposed multifunctional key management framework assures both node-to-sink and node-to-node
authentication along report forwarding routes. Moreover, our data delivery approach guarantees efficient en-route bogus data filtering and is highly robust against DoS attacks.
We evaluate our design through extensive analysis, which demonstrates its high resilience against an increasing number of compromised nodes and effectiveness in energy savings that is, achieving 85 percent or more energy savings in contrast to the case without using our design when appropriate parameters are chosen

LEoNIDS: a Low-latency and Energy-efficient Network-level Intrusion Detection System

Over the past decade, design and implementation of low-power systems has received significant attention. Started with data centers and battery-operated mobile devices, it has recently branched to core network devices such as routers. However, this emerging need for low-power system design has not been studied for security systems, which are becoming increasingly important today. Towards this direction, we aim to reduce the power consumption of Network-level Intrusion Detection Systems (NIDS), which are used to improve the secure operation of modern computer networks. Unfortunately, traditional approaches to low-power system design, such as frequency scaling lead to a disproportionate increase in packet processing and queuing times. In this work, we show that this increase has a negative impact on the detection latency and impedes a timely reaction. To address this issue, we present Leones: an architecture that
resolves the energy-latency tradeoff by providing *both* low power consumption *and* low detection latency at the same time. The key idea is to identify the packets that are more likely to carry an attack and give them higher priority so as

to achieve low attack detection latency. Our results indicate that Leones consumes comparable power to a state-of-the-art low-power design, while, at the same time, achieving up to an order of magnitude faster attack detection.

Low power consumption has emerged as one of the main design goals in today's computer systems. Recently, much effort has been put into improving the energy efficiency in a variety of areas like data centers, high performance computing, mobile devices, and networks . Towards this direction, we aim to build an energy-efficient Network level Intrusion Detection System (NIDS). NIDS are commonly deployed to detect security violations, enhancing the secure operation of modern computer networks. They perform computationally heavy operations like pattern matching, regular expression matching, and other types of complex analysis to detect at real time malicious activities in the monitored network. Thus, NIDS usually utilize multi-core systems or cluster of servers to cope with increased link speeds and complicated analysis. However, the energy efficiency of security systems like NIDS has not received significant attention and has not been studied before. In this work, we study this emerging need for low-power system design and improved energy efficiency focusing on NIDS, which are among the most commonly deployed systems for cyber security. Although NIDS are usually provisioned to operate at link rate, in order to be able to handle a fully utilized network, most networks are typically much less utilized. This results in increased power consumption at low traffic load. To reduce the energy spent under low traffic we aim at building a power proportional NIDS using Dynamic Voltage and Frequency Scaling (DVFS) and sleep states (C-states), which can be found in modern processors. The system should consume the less power needed to sustain the incoming traffic load. We found that a NIDS consumes less power when it uses the smallest number of cores that can operate at the lowest possible frequency to process the network traffic, by keeping these cores nearly fully utilized. This energy-efficient NIDS can process all packets with up to 23% lower power consumption than the original system at low rates. However, we observe a significant increase on the detection latency due to higher processing times when reducing the frequency, and mostly due to increased queuing delays imposed by the high utilization. A low detection latency is very important to ensure a timely reaction to the attack. Upon the detection of a packet that carries an attack, the NIDS can actively terminate the offending connection or install a new firewall rule. This reaction should be immediate, before the attack packets reach the victim's machine and the attack succeeds. Therefore, our results indicate a new tradeoff for NIDS: the *energy-latency tradeoff*. Our key idea to resolve this tradeoff is to identify the most important packets for attack detection and process them with higher priority, resulting in low latency and fast detection. The rest packets are processed with lower priority to achieve overall low power consumption. We explore two alternative approaches to reduce the latency of high-priority packets: *time sharing* and *space sharing*.

In time sharing we use a typical priority queue scheduling in each core. In space sharing the high-priority packets follow a different path, using dedicated cores with much lower utilization to achieve low latency. To implement space sharing we use features of modern network interface cards (NIC) to move efficiently the processing of least-significant packets to cores with higher utilization, a technique we call as *flow migration*. We experimentally compare the two approaches and we find that space sharing has a better power-latency ratio. Based on these approaches we propose LEoNIDS: a NIDS architecture that resolves the energy-latency tradeoff. The implementation of LEoNIDS uses NIC features, a specialized kernel module, a modified user-level library, and it is based on the popular Snort NIDS.LEoNIDS consumes less power, proportionally to the traffic load, while its detection latency remains low and almost constant at any traffic load. The main contributions of this work are,

• We identify a new tradeoff for NIDS: the *energy-latency tradeoff*. As we reduce power consumption, the detection latency is significantly increased, which impedes a timely reaction to incoming attacks. We found that the main cause of this increase is the queuing delays imposed by the high core utilization.

• We resolve the energy-latency tradeoff by identifying the packets that have a higher probability to contain an attack and processing them with higher priority.

• We introduce *space sharing*: a new technique based on flow migration that processes high-priority packets in dedicated cores with low utilization, and moves the low priority packets to cores with higher utilization.

• We experimentally compare two alternative approaches for low latency in a power-proportional NIDS. We show that space sharing results in lower detection latency when power consumption is reduced.

• We present the design, implementation, and evaluation of LEoNIDS, a NIDS architecture that achieves both low latency and reduced power consumption.

In this work we studied the problem of improving the energy efficiency of NIDS using common power management capabilities like DVFS and C-states. First, we identified an energy latency tradeoff: the reduced power consumption results in a significant increase of the detection latency, which impedes a timely reaction of NIDS to incoming attacks. We showed that the main reason of this increase is the high queuing delays imposed by high core utilization. Then, we presented the design, implementation, and evaluation of LEoNIDS: a NIDS that resolves the energy-latency tradeoff. They key idea of LEoNIDS is to process with higher priority the first few bytes of each flow, which have a higher probability to carry an attack, to achieve lower latency for them and faster attack detection. We proposed two alternative techniques: time sharing, which uses a typical priority queue scheduling, and space sharing, which uses dedicated cores with low utilization to process high-priority packets. Our experimental evaluation shows that LEoNIDS performs better with space sharing, resulting in low power consumption, proportionally to the load, and constantly low attack detection latency at the same time

Architectural Support for Fast Symmetric-Key Cryptography

The emergence of the Internet as a trusted medium for commerce and communication has made cryptography an essential component of modern information systems. Cryptography provides the mechanisms necessary to implement accountability, accuracy, and confidentiality in communication. As demands for secure communication bandwidth grow, efficient cryptographic processing will become increasingly vital to good system performance. In this paper, we explore techniques to improve the performance of symmetric key cipher algorithms. Eight popular strong encryption algorithms are examined in detail. Analysis reveals the algorithms are computationally complex and contain little parallelism. Overall throughput on a high-end microprocessor is quite poor, a 600 Mhz processor is incapable of saturating a T3 communication line with 3DES (triple DES) encrypted data. We introduce new instructions that improve the efficiency of the analyzed algorithms. Our approach adds instruction set support for fast substitutions, general permutations, rotates, and modular arithmetic. Performance analysis of the optimized ciphers shows an overall speedup of 59% over a baseline machine with rotate instructions and 74% speedup over a baseline without rotates. Even higher speedups are demonstrated with optimized substitutions (SBOXes) and additional functional unit resources. Our analyses of the original and optimized algorithms suggest future directions for the design of high-performance programmable cryptographic processors.

INTRODUCTION

In an increasingly connected world, cryptography has become an essential component of modern information systems. Cryptography provides the mechanisms necessary to provide accountability, accuracy and confidentiality inherently public communication mediums such as the Internet. Today, cryptographic processing is primarily reserved for electronic commerce transactions and secure e-mail, however, the adoption of virtual private networks (VPNs) and secure IP (IPSEC) will subject more of all communication to cryptographic processing. As secure communication bandwidth demands continue to grow, so too will the importance of efficient cryptographic processing. Cryptography is the art of using mathematics to encrypt and decrypt data. There are many cryptography algorithms (or ciphers) in use today, some good, and some not so good. The quality of a cipher is judged by its ability to prevent an unrelated party from determining the original content of an encrypted message. The simplest ciphers are known as *symmetric-key ciphers*. Communicating parties share a common private key which is used to transform the message from *plaintext* to *ciphertext*. The ciphertext is communicated to the other party, and then the process is reversed using the same private key. The primary obstacle in making private key symmetric ciphers useful is distribution of private keys. To securely share a private key, communicating parties would first have to be holding a shared private key! Public key cryptography solves this conundrum by implementing encryption with two keys, a well-known public key and a private key. Only the receiver knows the private key value. The receiver's public key, on the other hand, is widely published by trusted sources. The resulting ciphertext can only be decrypted using the receiver's private key. Secure communication now proceeds without any insecure exchanges of private information. The process may also be reversed to produce what is known as a *digital signature*. Digital signatures *authenticate* the sender, *i.e.*, verify the identity of the sender. Since only the person holding the private key knows its value, only that person can create a digital signature that others can decrypt with the public key (assuming the private key has not been compromised). It would seem that the additional benefits of public key encryption would obviate the need for private key encryption, however, the high cost of employing public key encryption requires that it be used sparingly. Strong public
key ciphers are computationally very expensive,

runningClearly, for very short sessions fast public key cipher processing is crucial for high transaction throughput. For a 32k session length, private key processing overheads rise to 48% of overall run time. Since a session will likely see a user visit many web pages and web pages with many objects, private key cipher performance will quickly dominate the performance of SSL sessions. To further reduce the cost of public key authentication, SSL allows the use of a session cache, where authenticated private keys are held and can later be reused when users reconnect to view other web pages. In this paper, we focus our attention on improving the performance of private key symmetric ciphers. We first examine the execution of eight widely known strong symmetric-key ciphers. We analyze their performance on detailed micro architectural models, where we are able to clearly show their performance and the bottlenecks that slow their progress. Armed with these insights, we propose architectural extensions that streamline cipher kernel processing. The new instructions speed modular arithmetic, substitutions, general bit permutations, and rotates. We recode the cipher using these new instructions and then examine their performance on micro architectural models with varying levels of support for fast cryptography. Our approach is a general one, the instructions are shown to be useful across a broad array of cipher algorithms.

CONCLUSIONS

As the Internet moves to the forefront as a trusted medium for commerce and communication, cryptography has become a integral part of modern information systems. We showed that even on very high-end microprocessors, common cryptographic algorithms (such as 3DES) cannot produce the throughput necessary to fully saturate a single T3 communication line - large web sites will often service many of these lines simultaneously. Furthermore, as the Internet and its applications move toward more security in communication, cryptographic bottlenecks will continue to grow. We present detailed analyses of eight popular private
key symmetric ciphers. We find that their performance and setup times vary widely. Analysis of their bottlenecks reveals that many of the algorithms run at near dataflow speed, those that do not simply require more resources for additional performance. Given these analyses, we proposed new instructions that speed the common operations of symmetric ciphers. Instruction set support is added for substitutions, permutations, rotates, and modular multiplication. We then examine their performance on micro architecture models of varying cost and performance. Performance analysis of the optimized benchmarks revealed an 59% speedup over machines with rotate instructions, and a 74% speedup over machines without rotates. Our analyses of the original and optimized algorithms suggest that there is more opportunity to improve the performance of cryptographic processing. Currently, we are exploring other optimizations for the eight kernels presented in this paper, including optimization of their setup routines and faster permutations. Looking further out, we are also exploring the possibility of hardware-cipher code signs. Are there efficient functions that could be mapped to a processor pipeline what would at the same time be fast and have good diffusion properties? If so, it would make the case for a hardware-cipher co-design. In this paper, we proposed techniques to add fast cryptography support to a general purpose processor. We are now exploring the implications of a design where the primary purpose of processor is cryptographic processing. Cryptographic processors would have to deliver orders of magnitude more performance to meet the bandwidth demands of secure servers and virtual private network (VPN) routers. This is quite an interesting design space: the processor must have general capabilities so that it can support a wide array of (possibly yet-to-be-invented) cryptographic ciphers, but it need not support the generality of all programs. SPEC performance is irrelevant for these processors; they need only execute cipher kernels quickly (both private and public key algorithms). Optimizations we are considering include fine-grained multi-threaded micro architectures to extract inter-session parallelism, four operand instructions to permit increased operation combining, and micro architecture-based loop optimizations. We are currently exploring these designs in depth and will report on their design and evaluation in a future paper.

A Security Punctuation Framework for Enforcing Access Control on Streaming Data

The management of privacy and security in the context of data stream management systems (DSMS) remains largely an unaddressed problem to date. Unlike in traditional DBMSs where access control policies are persistently stored on the server and tend to remain stable, in streaming applications the contexts and with them the access control policies on the real-time data may rapidly change. A person entering a casino may want to immediately block others from knowing his current whereabouts. We thus propose a novel "stream-centric" approach, where security restrictions are not persistently stored on the DSMS server, but rather streamed together with the data. Here, the access control policies are expressed via security constraints (called security punctuations, or short, sps) and are embedded into

data streams. The advantages of the sp model include flexibility, dynamicity and speed of enforcement. DSMSs can adapt to not only data-related but also security-related selectivities, which helps reduce the waste of resources, when few subjects have access to data. We propose a security-aware query algebra and new equivalence rules together with cost estimations to guide the security-aware query plan optimization. We have implemented the sp framework in a real DSMS. Our experimental results show the validity and the performance advantages of our sp model as compared to alternative access control enforcement solutions for DSMSs.

A. Security in Data Streaming Environments The need for people to protect themselves and their assets is as old as humankind. The increasing use of electronic, sensor and GPS devices means that individuals today have an ever-growing range of electronic (data) assets that may potentially be at risk. When computing devices are integrated with people, various personal information is expressed in digital form. Devices can communicate this information over networks and users have no control over who and for what purpose may query their data. Some users, knowing that their personal information (e.g., location, health condition) is not safeguarded, may hesitate to use such devices because of the risk of data being misused.Traditional access control schemes which typically assume finite persistent datasets and static (or rarely changing) access control policies become largely inapplicable in this new stream paradigm. This inapplicability is due to the fact that stream environments tend to be highly dynamic. Data is continuously generated and may have different security sensitivities depending on the context, on personal preferences or on the streaming values - all of which may frequently change at a possibly very fine granularity.

B. Motivating Examples

Example 1: Protection against context-aware spam. People may want to block unwanted businesses from sending them advertisements based on their location or any other information. As a person is driving or walking, the device may adapt security constraints based on the proximity of the businesses and his/her preferences limiting to who would be allowed to "see" the person. This helps to impede focused marketing efforts and to avoid receiving "context-aware spam" – services or information people don't know of or agree to.

Example 2: Privacy protection of personal health data. A patient may be living at home with a health monitoring device attached to him which can detect early health abnormalities and transmit alert signals to relevant personnel. However, the patient may prefer only certain users, such as only his doctor or a nurse, to have access to his streaming data and prevent access for any third-parties (e.g., insurance companies or other hospitals). Only if his vital signs go far above the norm and he is in an imminent danger needing urgent care, should the closest hospital or ER gain access to his streaming data. We envision that individual devices transmitting streaming data will be able to inject their respective security restrictions together with the data. The policies, as will be illustrated, can be encoded into a compact format, and in most cases can be included into the same network message with the data. Thus little demand for additional network communication is expected. In this paper, we assume that streaming data is transmitted securely from a data source to the streaming database. That is, the possibility of the data being intercepted and compromised on the network is beyond the scope of this paper.

C. Alternative Access Control Mechanisms

To motivate our approach, we sketch and compare alternate methods to enforce access control on streaming data. Non-streaming: Store-and-probe approach. The policies on the streaming data are collected in one place and stored in a persistent table. Whenever an access to the data is requested, the policy table is probed, to see if the access should be granted or denied. The advantage of this approach is its simplicity. All policies are stored and updated in one place. The main disadvantage is its inability to cope with frequent policies' changes. Every change in a policy would require an update to the policy table, and every request to a data would require a look up to this central place. Large number of data sources, fine-granularity of policies, frequent policy changes and continuous lookups may create a bottleneck in the performance of a streaming system. Streaming: Tuple-embedded approach. An alternative mechanism is to stream security restrictions embedded inside data tuples. Different attributes in a tuple, however, may each have a distinct policy, which may lead to an explosion in tuple sizes, potentially seriously impacting the system performance. Moreover, tuples that arrive adjacent to each other in the stream are quite likely to be generated based on the same context (e.g., location, time), and thus may frequently have similar access control policies. Using this approach, tuples with identical policies would still carry their own (redundant) copy, and the query processor would still have to process every tuple individually to guarantee that no unauthorized access is granted. One possible improvement to minimize per tuple storage overhead could be to encode policies as bitmaps, and then abstract policy-based filtering using bitmap operations .A

bitmap representation is highly compressible, so the storage overhead would be somewhat minimized. However, even with compression, still this approach suffers from redundant storage and unnecessary per-tuple processing overhead. Streaming: Punctuation-based approach. The third alternative, which we adopt in our work, is a punctuation-based approach, where security meta-data tuples are interleaved with the data tuples in the streams. Punctuation-based solution has several advantages over the above-mentioned approaches. First, the access control is dynamic and the speed of enforcement is fast, because security restrictions are streamed together with the data. Second, the security punctuations may be shared by multiple tuples that have similar policies. Thus no redundant copies of policies are stored, memory overhead is minimized and the security-related processing is shared. Policies in security punctuations can also be encoded in a bitmap format for compactness, thus further reducing securityrelated processing. For ease of readability, in the rest of the paper we present security punctuations in the alphanumeric format.

D. Our Proposed Solution:

SP Framework We proposes to stream security constraints called security punctuations (or short sps)', interleaved with the actual stream data describing an access control policy on the upcoming portion of the stream. A sp is a predicate that informs the stream processor of who has access when to which streaming data. Data sources emit sps based on the user specifications.

In our work, we distinguish between two types of users: (1) users providing the streaming data, termed data providers (short DPs), and (2) users querying the streaming data, termed query specifiers (short QSs). When query specifiers register continuous queries for execution, each query inherits the security restriction(s) associated with the query specifier registered to receive the results. When the stream data arrives to the server, the database engine examines the streaming data tuples' policy stored in the sps and checks if the queries conform to the policy, discarding the data that no query has access rights to. We assume that data providers and users querying the data use the same access control model.

Contributions

* We introduce a novel sp model that supports declarative access control specification and enforcement on real-time streaming data.

* We propose “security-aware" stream algebra, by enhancing the traditional algebra with security-aware extensions and new algebraic equivalence rules. * We present a pipelined execution model enabled by the security-aware algebra and describe security-aware query optimization employing the new algebraic rules and cost estimations in query plan rewriting.

* experimental analysis on a real DSMS CAPE shows that sp framework is superior to alternative access control mechanisms for streaming data in terms of both processing and memory.

we have proposed a novel approach to enforce access control in streaming environments using security punctuations. This work makes three important contributions: (1) a scheme for defining security semantics on streaming data; (2) a query processing and optimization framework aware and compliant with the security restrictions; (3) an implementation and investigation of the security mechanism and its effect on query processing. The significance of our experimental results is that we have shown that our sp framework, integrated as a part of query processing, has very low overhead and outperforms alternative approaches. In the future, we plan to explore incremental access control policies and runtime changes in subjects' role assignments and their effect on query processing.

PRESENT: An Ultra-Lightweight Block Cipher

With the establishment of the AES the need for new block ciphers has been greatly diminished; for almost all block cipher applications the AES is an excellent and preferred choice. However, despite recent implementation advances, the AES is not suitable for extremely constrained environments such as RFID tags and sensor networks. In this paper we describe an ultra-lightweight block cipher, present. Both security and hardware efficiency have been equally important during the design of the cipher and at 1570 GE, the hardware requirements for present are competitive with today’s leading compact stream ciphers.

One defining trend of this century’s IT landscape will be the extensive deployment of tiny computing devices. Not only will these devices feature routinely in consumer items, but they will form an integral part of a pervasive and unseen communication infrastructure. It is already recognized that such deployments bring a range of very particular security risks. Yet at the same time the cryptographic solutions, and particularly the cryptographic

primitives, we have at hand are unsatisfactory for extremely resource-constrained environments. In this paper we propose a new hardware-optimized block cipher that has been carefully designed with area and power constraints uppermost in our mind. Yet, at the same time, we have tried to avoid a compromise in security. In achieving this we have looked back at the pioneering work embodied in the DES and complemented this with features from the AES finalist candidate Serpent which demonstrated excellent performance in hardware. At this point it would be reasonable to ask why we might want to design a new block cipher. After all, it has become an "accepted" fact that stream ciphers are, potentially, more compact. But we note a couple of reasons why we might want to consider a compact block cipher. First, a block cipher is a versatile primitive and by running a block cipher in counter mode (say) we get a stream cipher. But second, and perhaps more importantly, the art of block cipher design seems to be a little better understood than that of stream ciphers. We suspect that a carefully designed block cipher could be a less risky undertaking than a newly designed stream cipher. Thus, we feel that a block cipher that requires similar hardware resources as a compact stream cipher could be of considerable interest. It is important to realise that in developing a new block cipher, particularly one with aggressive performance characteristics, we are not just looking for innovative implementation. Rather, the design and implementation of the cipher go hand-in-hand and this has revealed several fundamental limits and inherent contradictions. For instance, a given security level places lower bounds on the block length and key length. Just processing a 64-bit state with 80-bit key places fundamental lower limits on the amount of space we require. We also observe that hardware implementation particularly compact hardware implementation favours repetition. Even minor variations can have an unfortunate effect on the space required for an implementation. Yet, at the same time, the cryptanalyst also favours repetition and seeks mathematical structures that propagate easily across many rounds. How much simple, repetitive structure can we includewithout compromising its security?

In this paper we have described the new block cipher present. Our goal has been an ultra-lightweight cipher that offers a level of security commensurate with a 64-bit block size and an 80-bit key. Intriguingly present has implementation requirements similar to many compact stream ciphers. As such, we believe it to be of both theoretical and practical interest. Like all new proposals, we discourage the immediate deployment of present but strongly encourage its analysis

rekeying process never disrupts ongoing data streams and encryption/decryption. SEEN supports the source node authentication and shared key recovery without incurring additional overhead. Time based access control can effectively reduce communication overhead in a network.

FUTURE ENHANCEMENTS

There is scope for future development of this project. The world of computer fields is not static; it is always subject to be dynamic. The technology which is famous today becomes outdated the very next day. To keep abstract of technical improvements, the system may be further refined. So, it is not concluded. Yet it will improve with further enhancements.

In this project we have developed a security model to withstand bandwidth consumption in cloud oriented big data processing services. The future of this proposed framework will implement deduplication identification mechanism for flexible storage mechanism.

REFERENCES

[1] A. Arasu, et al. "STREAM: the stanford stream data manager
(demonstration description)." In *ACM SIGMOD international conference on Management of data*, pp. 665-665, ACM, 2003.

[2] H-S. Lim, Y-S. Moon and E. Bertino, "Provenance-based trustworthiness assessment in sensor networks." In *Seventh International Workshop on Data Management for Sensor Networks*, pp. 2-7. ACM, 2010.

[3] S. Sultana, G. Ghinita, E. Bertino and M. Shehab, "A lightweight secure provenance scheme or wireless sensor networks." In *18th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 101-108, 2012.

[4] R. A. Shaikh, S. Lee, M. AU Khan and Y. J. Song, "LSec: lightweight security protocol for distributed wireless sensor network." In *IFIP International Conference on Personal Wireless Communications*, pp. 367-377. Springer Berlin Heidelberg, 2006.

[5] G. Selimis et al., "A lightweight security scheme for wireless body area networks: design, energy evaluation and proposed microprocessor design." *Journal of medical systems*, vol. 35, no. 5, pp. 1289-1298, 2011.

[6] G. Selimis, at al. "Evaluation of 90 nm 6 T-SRAM as Physical Unclonable Function for Secure Key Generation in Wireless Sensor Nodes", *in IEEE ISCAS Brazil*, pp.

567-570, 2011.
[7] M. Roesch,"Snort: Lightweight Intrusion Detection for Networks." *LISA*, vol. 99, no. 1, pp. 229-238. 1999.

[8] N. Tsikoudis, A. Papadogiannakis and E. P. Markatos, "LEoNIDS: a Low-latency and Energy-efficient Network-level Intrusion Detection System." *IEEE Transactions on Emerging*
Topics in Computing, vol. 4, no. 1, pp.142-155, 2016.

[9] W. Lee and S. J. Stolfo, "Data Mining Approaches for Intrusion Detection." In *Usenix security*. 1998.

[10] Y. Xie, D. Feng, Z. Tan and J. Zhou, "Unifying intrusion detection and forensic analysis via provenance awareness." *Future Generation Computer Systems*, vol. 61, pp.26- 36, 2016.
[11] A. S. Wander, N. Gura, H. Eberle, V. Gupta and S. C. Shantz, "Energy analysis of public-key cryptography for wireless sensor networks." In *Third IEEE international conference on pervasive computing and communications*, pp. 324-328. IEEE, 2005.

[12] T. Park and K. G. Shin, "LiSP: A lightweight security protocol for wireless sensor networks." *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 3, no. 3, pp. 634-660, 2004.

[13] A. Bogdanov, et al. "PRESENT: An ultra-lightweight block cipher." In *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 450-466, 2007.
[14] T. A. Zia and A. Y. Zomaya, "A Lightweight Security Framework for Wireless Sensor Networks." *JoWUA*, vol. 2, no. 3, pp. 53-73, 2011.

[15] K. Van Laerhoven, "Combining the self-organizing map and kmeans clustering for on-line classification of sensor data." In *International Conference on Artificial Neural Networks*, pp. 464- 469. Springer Berlin Heidelberg, 2001.

[16] P. Ferreira and P. Alves, *Distributed context-aware systems*. Springer, 2014. DOI 10.1007/978-3-319-04882-6

[17] D. Ganesan, D. Estrin and J. Heidemann, "DIMENSIONS: Why do we need a new data handling architecture for sensor networks?." *ACM SIGCOMM Computer Communication Review, vol.* 33, no. 1. pp. 143-148, 2003.

[18] D. Puthal, S. Nepal, R. Ranjan and J. Chen, "A Secure Big Data Stream Analytics Framework for Disaster Management on the Cloud." In *18th International Conference on High Performance Computing and Communications*, pp. 1218-1225. 2016.

[19] D. Puthal, S. Nepal, R. Ranjan and J. Chen, "A dynamic prime number based efficient security mechanism for big sensing data streams." *Journal of Computer and System Sciences* vol. 83, no. 1, pp. 22-24, 2017

[20] D. Puthal, S. Nepal, R. Ranjan and J. Chen, "DLSeF: A Dynamic Key Length based Efficient Real-Time Security Verification Model for Big Data Stream." *ACM Transactions on Embedded Computing Systems*, vol. 16, no. 2, pp. 51, 2017.