Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. V (2010), No. 5, pp. 961-967

Secure Data Retention of Call Detail Records

F. Vancea, C. Vancea, D. Popescu, D. Zmaranda, G. Gabor

Florin Vancea, Codruţa Vancea, Daniela Elena Popescu
Doina Zmaranda, Gianina Gabor
University of Oradea, Romania
410087 Oradea, St. Universităţii 1
E-mail: {fvancea,cvancea,depopescu,zdoina,gianina}@uoradea.ro

Abstract: In today's world communication is relying heavily on electronic means, both for voice and other native data. All these communication sessions leave behind journaling information by the very nature of the underlying services. This information is both sensitive with respect to user's rights and important for law enforcement purposes, so proper storage and retrieval techniques have to be taken into consideration. The paper discusses such techniques in relation with recent EU recommendations and suggests some methods for achieving good performance while preserving the required security levels.
Keywords: call detail records, secure retention, row chaining.

1 Introduction

During the last years we have seen an increasing interest in secure storage and retrieval of journaling archives of various transactions. The possibility of fraud or inappropriate usage of resources have led to laws and regulations regarding storage of transaction history for phone calls, SMS, e-mail traffic and other forms of communication. Specifically, the European Parliament and the Council of the European Union have adopted a directive (2006/24/EC) [1] which gives guidelines for retention of data collected from publicly available communication services or public communication networks, in order to ensure that the data is available for investigation, detection and prosecution of serious crime.

Usually, the devices that perform the actual communication function are generating logging data that can identify the peers, time attributes and other relevant details of the transaction. These logs have been traditionally used for billing and eventually for statistical purposes by the network operators themselves. Since this information was stored anyway by the operators, the law enforcement agencies could make use of them for specific purposes within a frame more or less regulated by each country laws. Currently, many states are requiring the voice and data operators to store these logs for a well-defined period of time and to present them timely when legally requested by various law-enforcement entities. In order for this information to be useful for investigation and furthermore for prosecution, there is an implicit requirement that the information contained by the logs cannot be tampered with.

Even if the operators should not (or actually may not) capture and store the actual content of the communication, there are also increasing concerns about the privacy of the service users related to their communication partners. Wholesale collection of user communication actions and habits can be viewed as a breach of privacy, so strong protection against casual browsing or unauthorized intentional usage of this information should be provided.

When combining the long-term storage and efficient retrieval requirements with privacy, non-repudiation, integrity, performance and cost requirements, the task of data retention becomes non-trivial.

2 System structure, main requirements and challenges

The communication network operator has to manage during normal operation several pieces of information about each communication session. At the bare minimum this information is used to establish communication channels and to ensure correct traffic flow but it may also be stored and used for billing purposes. We will call such information about one user session Call Detail Record (CDR) as this is the name commonly used by voice service providers. For simplicity we will also name the communication session call but all reasoning below applies also for data sessions (e-mail, Internet sessions).

The records originate in the network equipment (central switch, access infrastructure), which have limited ability to store data, so periodically the records are transferred into database systems where they are processed (e.g. for billing) and finally discarded. If CDR retention is desired, the records are also stored for a rather extended period of time in some sort of semi-persistent storage until lawfully required through a query or until the retention period expires.

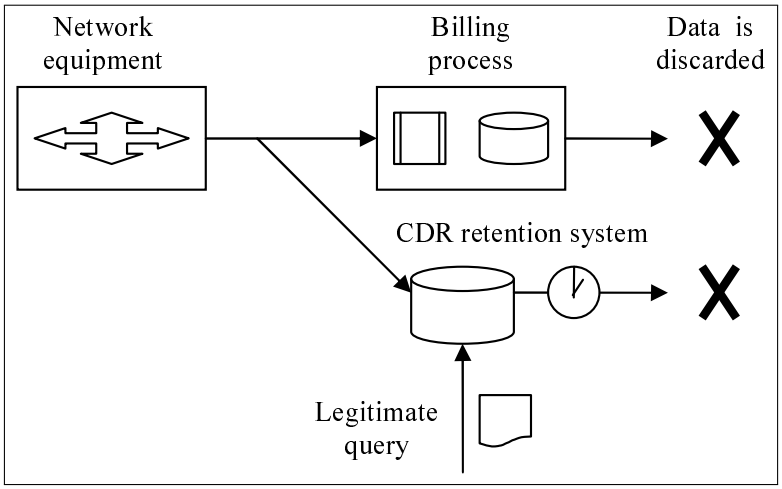

Figure 1: CDR data flow and data retention

For the purpose of this discussion we will not consider the billing path because this is part of the existing business model of the operator and is already well-established and less likely to accept any change. User data may be exposed there but this is an old threat, which is supposedly already handled by operator's procedures. Furthermore, the lifecycle of user sensitive data there is relatively short (one month or at most a couple of months) and the impact of integrity loss is only eventually financial to the operator but not deeply legal to the user.

The CDR data can be attacked:

- Before entering the retention system;
- While stored in the retention system;
- During query;
- After supposed deletion.

We will consider that the attacks performed before entry in the retention system have a low risk because the data stays in transit between network equipment and storage for a short period of time, the path is internal to the operator (thus is reasonably secure) and more important, the attacks are likely to take place only after a reasonable long time after the call is placed (otherwise the attacker would have refrained from making the call at all or would have attacked directly the network infrastructure to obtain an untraceable call).

The attacks on CDR data during the query may be:

- Impersonating authorized entities;
- Query alteration;
- Result alteration;
- Result disclosure.

All those can be stopped by proper signing and encryption of the request and the result.

The attacks after supposed deletion are disclosure attacks and would attempt to extract protected information from media used to store CDR data in the retention system. Encrypting the data is of course the main protection method but ciphertext-extensive attacks may exist which exploit the very large amount of encrypted data. To avoid these attacks write-once media should be destroyed properly and rewritable media should be properly erased after retention time expires.

We will focus on data stored in the retention system, because the CDR data stays there a rather long time (at least 6 months, according to EU recommendation) and the attacks may be either against the confidentiality or against the integrity of the data.

The CDR retention system has to take steps to prevent at least the following attack types:

- unauthorized disclosure of call detail, either for a particular call, a particular originator or a particular termination
- alteration of call detail (originator, termination, start/end time, other details)
- complete removal of one call or of all calls matching a particular originator, termination or time interval
- complete denial of service against the retention system

When evaluation the potential solutions we should also consider the size of the problem. One reasonably large operator may easily originate or terminate in excess of 10M calls per day (probably at least one order of magnitude more in the case of data services). At a minimum, a CDR will contain the originating subscriber number or identifier (at least 10 characters), termination or forwarding subscriber number (at least 10 characters), starting timestamp and duration (about 6 characters). To cover all situations (international dial numbers) additional space has to be reserved. In the case of mobile services additional location data is required and data services may have longer identification tokens. We estimate therefore the raw CDR to require at least 32 bytes of storage, leading to a minimum figure of 320MB of raw data to be stored for each day of traffic. This extends to about 10GB monthly and 120 GB yearly.

The above figures may appear small by today's standards in memory and storage capacity, but they are absolute minimum estimates of raw data. In a working system there should be protection which is obviously introduced by encryption and a working system should offer some sort of direct random access to the data, preferably optimized like an indexed database query. Both features have storage overhead and we consider it to be tenfold from extrapolating previous experiences. In the end the retention system will probably have to deal securely with about 500GB of data for a 6 month retention period on the assumption of 10M calls per day.

The above estimation may still appear small by comparison to a large business platform, but we should keep in mind that the CDR retention system is a non-profitable investment for the operator so one cannot expect it to use significant resources in terms of purchased hardware or software. Fortunately, the criteria for common queries are limited: by originator identifier (directly or indirectly mapped through user identity), by termination identifier and by time interval.

3 Potential solutions

According to the problem description we should figure how to store the data and how to efficiently retrieve the required CDR according to the typical query criteria.

3.1 Direct WORM storage

One obvious approach is to use WORM media, given that the write-once feature will provide tampering protection. This approach will not automatically solve the confidentiality issue, so some additional measures have to be taken. The main disadvantage of passive WORM media is capacity. Considering the above estimations, plain passive WORM media is not sufficient unless the time span for data written on one support is short. The time span should also be as short as possible, because data is protected only after write, and data in transit will be vulnerable to integrity attacks. Short time spans on one support will translate into many media units for the entire retention period, making the query process difficult and time-consuming. The speed of passive WORM media is also limited, affecting the query performance.

A better performing approach is to use special active WORM devices. These are special storage units presenting the WORM feature at the access port (Ethernet, SCSI) but backed by conventional magnetic storage. The WORM feature is protected by dedicated interface and firmware and the whole device is tamper-proof. This kind of device offers good protection but is usually expensive and its main use case is append to log. Our goal is to map database-like search capabilities over the protected data and the append to log restricted primitive is not friendly with common database engines and their storage management.

The lesser version of active WORM is a hybrid software system that achieves the write once feature by modification of the operating system block drivers or by a custom operating system [2]. The protection it provides may be reasonable but far weaker than the one offered by true passive optical media. This hybrid alternative fails to be database-friendly, too.

All the solutions based mainly on WORM devices will amount to periodically storing the CDR logs and sequentially scanning those for each query. This is not very efficient; especially considering that all the data that is sequentially scanned will be encrypted for confidentiality purposes.

3.2 Database backed by WORM storage

A slightly better solution would be to use a true database for query purposes and a passive WORM reference for result validation.

The CDR data would both be stored in the database and consolidated with a reasonable short interval to WORM storage (passive media should be OK). The initial query would be performed over the regular database and the integrity of the database itself might be checked against some hash checks stored on WORM media. The solution is fragile and may not provide either the required level of integrity or the desired level of performance.

Some DBMS (e.g. Oracle) offer the possibility to split the storage segments over read-write and read-only media. In such a simple scenario, the retention time window would be split in several fragments out of which only the last one is residing on read-write storage. Of course, management of splitting would be rather complicated and there are certain limitations in place which would limit the reasonable unsafe fragment to at least one week. The solution has the disadvantage of poor query performance for time frames located on read-only media, but the main disadvantage is that the unsafe fragment cannot be protected even if conventional row-level encryption is applied. Little can be done to actually prevent the rows in the read-write partition from being maliciously created, changed or deleted. Security mechanisms at database

level do exist but we are looking for an additional layer of trust, which would rule out any database authority from tampering the data.

By using encryption at row level one can simply prevent row changing and unauthorized row creation [3]. Efficient query is possible by index-scan over plaintext time attributes or by index-scan over pre-encrypted source or target fields. Unfortunately row deletion can be always performed at some authority level without being even detected.

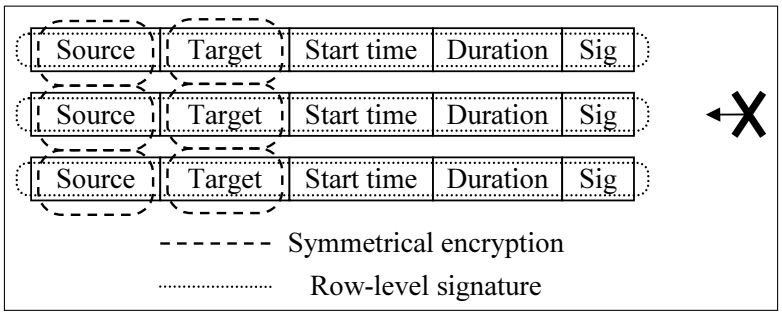

Figure 2: Row content is protected but rows can be deleted

3.3 Database with cross-linked rows backed by WORM storage

An improved solution is using results of our previous work, namely row chaining [4] [5]. Beside symmetrical encryption for confidentiality protection and row-level signature for row integrity and authenticity protection we use a chaining scheme that links one row to the previous one. The scheme can detect row deletion unless a whole block of rows from the end are collectively deleted. This would be solved by building the chain two ways, at the cost of revisiting previous records. Given the volume and the dynamic of data we consider a single chain is providing enough protection.

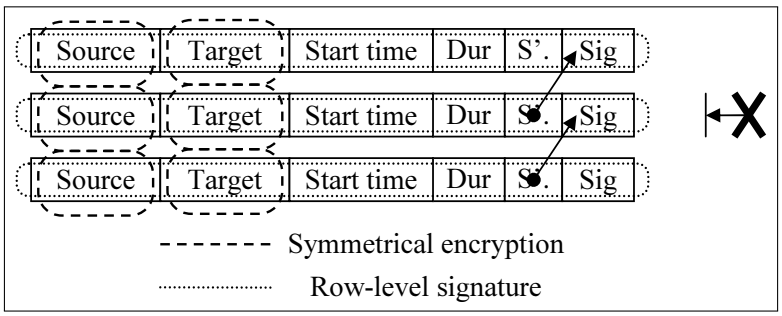

Figure 3: Rows protected by chaining cannot be deleted

Using the chaining scheme has the great advantage of relieving us from calling the WORM storage upon each query. The CDR data can now reside entirely in plain read-write database storage as its integrity is now protected by chains and signatures. When a query is presented, the results are quickly determined at regular database speed, and then the result-set is validated by walking the chain one or two steps for each record. The WORM storage is still required in case tampering is detected in any form and as a backup to avoid denial-of-service attacks (complete corruption of the database).

The above scheme has however a flaw and can be improved. One can maliciously alter the database indexes to skip certain records, and those records will never be returned in the result

set even if they are present in the database and the storage structure is still cryptographically intact. All checks will succeed, even for queries returning rows adjacent to the masked ones.

The solution is to build the chains not on subsequent rows as they are added to the database but on rows belonging to the same source or the same target. This means that the database will contain a chain of rows for each source and a chain of rows for each target. There is a new efficiency issue here, because we will need two chaining fields, one for source and one for target. The issue of protecting the last record in the chain becomes also apparently more sensitive, because the open ends of the chains are living longer. However, undetected deletion of a row requires that it is at the open end of both source and target chains, which is unlikely to happen for long in regular traffic.

4 Aggregation attack and countermeasures

So far we have considered that encrypting the source and the target fields offers appropriate protection against disclosure of the actual participants to a call. Unfortunately, in order to benefit from indexed search a particular source or target identifier has to be encrypted always with the same key, yielding the same encrypted value. The key does not have to be the same for all encryptions (actually should be changed over the source-target population) but once an identifier is encrypted in the system it should use the same ciphertext representation.

This invariance property required for indexing opens a path for aggregation attacks. Say the attacker has access to the database (either online or in archived form) and is looking for source A, but does not know the key. Actually it does not need to. If the attacker knows that A has placed calls at some particular moments in time it can aggregate through the data looking for that particular call pattern. Once it finds it all the past and future CDR involving A (at least as source) are accessible. The required number of such fixed points is not very high, as it was previously proven in aggregation attacks on blinded statistical data. The success of the attack depends on the traffic volume around the known time-points and on the number of the time-points available. The attacker may even optimize its chances by explicitly placing calls to A at low-traffic time intervals or if the attacker is itself A, by making calls at such well-chosen moments.

To provide some level of protection against this kind of attack we have to dilute the resolution the attacker has in the time domain when building a candidate set based on a given timestamp. When the attacker has a timestamp it can effectively build a set of sources or destinations that have placed or received a call at that moment in time. If the timestamp is say 11:00:00 AM, the set will be likely large (high traffic at busy hours) but if the timestamp is 03:00:00 AM the set will be considerably smaller. The time resolution dilution has to be made by sacrificing performance at some level. We will store in the Start time field a down-rounded timestamp and in an additional encrypted field the offset of the actual start time. This will reduce the resolution of the time searches and will entail some sequential processing to achieve exact query results. However, since the query specifies typically also the source or the target the performance penalty will be small (will use source or target indexes). On the other hand, the attacker has only the degraded start time to work with. To further refine the protection, the rounding should be made according to traffic patterns. Lower traffic intervals should be larger and higher traffic intervals may be smaller. This helps reducing the performance degradation we mentioned before and can be achieved by a Hamming-like algorithm applied once in a while to determine the optimal rounding intervals.

Even when using the rounding method to protect against data aggregation, special care should be taken with respect to the data storage pattern. All records have to be added incrementally with the starting timestamp (or eventually the end timestamp) because that is the way they come

from the network device logs. If the rounding algorithm performs for example the rounding at 23:00, 01:00, 05:00 the attacker may place a call shortly after 01:00 AM and look for records after but in close proximity of the Start time switch from 23:00 to 01:00. The solution is to randomly shift the rounding boundaries each day with at least 10% of the theoretical interval size.

5 Conclusions and Future Works

Designing and implementing a system which provides secure data retention of Call Detail Records is not a trivial task. There are several inter-related issues ranging from storage methods to data protection by encryption, and from efficient query to safe implementation. Unfortunately we have to note that in the current form, secure data retention has little driving force. The operators that manage the data have little to gain by properly implementing such a system and proper implementation is not necessarily cheap. The laws mandating the presence of such systems are rather vague on the required security and performance levels for such systems. The potential direct users have no direct means to encourage proper (read fast) implementations and the real owners of the call data (the subscribers) have also little means to encourage proper (read secure) implementations.

We however tried to outline within the limited size of this paper some of the requirements and the challenges related to such a system together with means to achieve a safe system with good performance.

Of course, each link in the chain has its very important contribution to the overall strength and there are many details left uncovered. Some directions may be: the proper protocol to be implemented between the retention system and the authorized partner requesting a query, proper and efficient protocol for destruction of CDR data at the end of retention period, implementation details on the chaining and on the time dilution algorithm, general implementation notes regarding secure split-role administration. Some of our future work may be related to these directions.

Bibliography

[1] ***, Directive 2006/24/EC of the European Parliament and of the Council Of 15 March 2006, *Official Journal of the European Union*, L105/54, 13.04.2006

[2] Y.Wang, Y.Zheng, Fast and secure WORM storage systems, *Proceedings of the IEEE Security in Storage Workshop (SISW)*, pages 11-19, 2003

[3] B. Schneier, Applied Cryptography, *John Wiley & Sons*, 1996.

[4] F. Vancea, C. Vancea, F. Vancea, C. Vancea, Practical Security Issues for a Real Case Application, *Int. J. of Computers, Communication and Control 3(S)*, pages 11-16, 2008

[5] F. Vancea, C. Vancea, Protecting data integrity with chained rows and public key cryptography. Comments on a real case, *Proceedings of RSEE 2008*