The least privacy-damaging centralised traffic data retention architecture (Extended abstract)

George Danezis

Microsoft Research Cambridge,
Roger Needham Building,
7 J J Thomson Avenue,
Cambridge, CB3 0FB, U.K.
gdane@microsoft.com

Abstract. We present a protocol that can be used by service providers to store traffic records centrally, while only making them readable to law enforcement after a proper authorisation has been issued and logged. Despite the system's best efforts to prevent mass profiling and surveillance, we discuss how it is inherently fragile.

1 Introduction

On 15 March 2006 the European Union adopted Directive 2006/24/EC requiring member states to ensure traffic data is retained and made available to law enforcement for periods ranging from 6 months to 2 years after they were generated[1]. The classes of traffic data to be retained are rather specific, and include means to identify the communicating parties, the duration and time of a communication, the devices used and their location. The directive was fiercely opposed by privacy advocates and a couple of countries are challenging it at the European Court of Justice.

Unsurprisingly [EPH04,Tay06], the UK government is considering the option of implementing a centralised repository of traffic data, to hold the records currently stored and managed at telecommunications and internet service providers. The cost of this project is rumoured to be tens of billions of pounds[2], and is justified on the basis of faster response times to law enforcement requests, reduced costs for service providers, and more efficient operations overall.

Traffic data retention itself, and the centralisation of traffic data are already creating systemic and large scale risks when it comes to the privacy of communication. To add to this problem, a naive implementation of a centralised scheme, where data would simply be stored on a data centre with a simple reference

[1] "Telecommunications Data Retention", Wikipedia, http://en.wikipedia.org/wiki/Telecommunications_data_retention

[2] "UK.gov to spend hundreds of millions on snooping silo", The Register, http://www.theregister.co.uk/2008/08/19/ukgov_uber_database/

monitor authorising and servicing requests, would enable wide spread abuse of the stored data.

The aim of this work is to propose an architecture that would minimise the potential for large scale abuse and data-mining operations of a centralised traffic data store. It also enables some auditing of the uses made of the database, without increasing, for technical reasons, the delays necessary to serve requests for data. Even our approach has drawback, and it does not totally neutralise the risks inherent in blanket centralised retention of traffic data.

The design philosophy behind our architecture is that *technical bottlenecks* that slow down the servicing of law enforcement requests, are removed by maintaining a central store of traffic data. This cuts down on the storage, retrieval and transfer costs of the service providers, and ensure the records are present when needed by law enforcement. On the other hand the procedure requiring the *collaboration of service providers* for the data to become legible are maintained in place, and augmented to ensure that a proper audit trail of accesses can be reconstructed. This approach somehow "balances" the need for high integrity and availability of the data for LE, reduced costs for service providers, and some privacy guarantees against blanket surveillance and data mining for users.

As we discuss, even the proposed scheme is easier to abuse than a decentralised data retention regime, or even better a data preservation regime. Another drawback, is its higher cost over a solution that does not protect privacy using cryptography, but only though procedural control, that are trivial for insiders to bypass. The fraction of the budget devoted to increasing surveillance versus ensuring privacy will be a lasting testament to the relative importance policy makers attach to these two features of the system.

2 A secure remote record storage scheme

The architecture proposed for the safe centralised storage of traffic data is based on the literature on secure logging [SK99] as well as secure remote storage [GSMB03,LKMS04]. The principle behind our design is that a central store of encrypted data is maintained centrally, yet the keys to identify or decode particular records are derivable only by the service provider. Furthermore the key derivation requires the LE to disclose some information about the target and the scope of the data being accessed, that can be logged.

The storage and retrieval protocol works in 3 stages:

1. **Storage.** The service provider creates a record, derives a key, and sends the encrypted record as well as an index to the central repository of traffic data.
2. **Request.** The law enforcement authority makes a request to the service provider, to get access to records with a particular target identifier, time window and type.
3. **Response.** After logging the request the service provider derives the keys necessary to locate and decode the records and sends them to the law enforcement authority.

We first describe the key hierarchy used to encrypt and identify each record, and then the protocols necessary to reveal parts of it to law enforcement.

2.1 A key hierarchy for traffic data records

We assume that each service provider has an identity and generates a set of traffic data records $R = \{R_0, \ldots, R_i, \ldots\}$. We are not overly concerned with the structure of each record R_i but we assume there are some deterministic procedures to extract information from each of them:

- provider(R_i), extract the identity of the provider that generated the record.
- type(R_i), extracts a set of types for the record.
- subject(R_i), extracts a set of network identifiers that the record relates to, annotated by their protocols.
- time(R_i), extracts a set of time periods related to the record. It is up to the system designers to decide the granularity of the time period, but in this work we assume to have a resolution of about a day.

Each provider has a sequence of symmetric master secret keys associated with the tuple of provider ID and time, that we denote $\mathcal{K}_{P \times t}$, where $\mathcal{P}$ is the provider and t is the time window. Given a record R_i we extract the four-tuples consisting of the Cartesian product of its provider, types, subjects and times, that we denote as:

$$L \equiv (P_j, T_j, S_j, t_j) \equiv \{\text{provider}(R_i)\} \times \text{type}(R_i) \times \text{subject}(R_i) \times \text{time}(R_i). \quad (1)$$

We assume that the function $H(\cdot)$ acts as a perfect hash function and random oracle. Given the set L of four-tupes for a record K_i we derive $|L|$ separate keys for each combination of attributes. Symmetric key $K_j = H(K_{P_j,t_j}, P_j, T_j, S_j, t_j)$ corresponds to the four-tuple (P_j, T_j, S_j, t_j). Furthermore multiple symmetric keys can be associated with each unique combination of attributes, by using a counter and deriving a further key $K_{jk} = H(K_j, k)$, where $k = 0, 1, 2, \ldots$

We denote as keys(R_i) a function that given a record R_i extracts all its relevant attributes (P_j, T_j, S_j, t_j), and returns all corresponding keys K_{jk}.

2.2 Cryptographic packaging of records

Each record is encrypted and tagged individually as soon as it is created. The encryption ensures that a key is necessary to recover the record in full, and the ID tag facilitates the recovery of specific records within a database. We assume a secure randomised symmetric encryption method that we denote as $E_K(\cdot)$, where K is the symmetric key. The corresponding decryption operation is denoted as $D_K(\cdot)$.

Given a record R_i we extract all keys corresponding to its combination of attributes K_j as described in the previous section. The sequence number k is chosen such that no other record was encrypted under the same k, the key $K_{jk} = H(K_j, k)$ is derived and the counter k is increased.

The encrypted record takes the form:

$$\mathrm{ID}_{R_j} \equiv H(\text{“ID”}, K_{jk}); \mathrm{BODY}_{R_j} \equiv E_{K_{jk}}(R_j) \tag{2}$$

The first component ID_{R_j} is the tag of the record, and the second component BODY_{R_j} is the body.

Given a database of N tuples of tags and bodies, as well as a key K_j it is possible to locate all records in time proportional to $\mathcal{O}(k \cdot \log N)$. The sequence of keys K_{jk}, starting with key $k = 0$ is generated, and for each ID $H(\text{“ID”}, K_{jk})$ a $\mathcal{O}(\log N)$ algorithm can be used to look-up the corresponding record. When the first k is tried for which the record does not exist, the search terminates.

It is worth noting some security properties afforded by this encoding:

- Trivially, given an encrypted record it is unfeasible to decrypt it without the appropriate key.
- Given two encrypted records it is unfeasible to tell whether they have been encrypted under the same key K_j or not, unless that key is known. The IDs are computed using the derived keys K_{jk} containing a different value of k.
- Given a key K_j it is easy $(k \cdot \mathcal{O}(\log N))$ to locate all records encrypted with this key.

These properties are important in designing the protocol to securely store and efficiently decode the encrypted records.

2.3 A remote storage and logged access protocol

The justification for centralised data retention is speed of access, and completeness of the data held. We design a protocol to ensure the data can be accessed quickly, as soon as an authorisation has been logged, while at the same time preventing access without authorisation and logging. In particular our scheme prevents the traffic data store from performing fishing expeditions, or profile without authorisation. Two simple protocols are needed for this: a *storage* protocol and an *access* protocol, executed between the service provider (SP) and a central data store (CDS).

Storage. The storage protocol simply consists of the service provider (SP) encrypting each new record R_i, as it is generated with all keys $\mathrm{keys}(R_j)$, and sending the resulting ciphertexts to the central data store (CDS).

Access (request-response). The authority managing the central data store can request access to some of the stored records. To do this it has to decide the criteria for selecting the records to access: namely a window of time as a set of periods t_{CDS}, the set of service providers concerned P_{CDS}, the set of subjects concerned S_{CDS}, and the types of records required T_{CDS}.

The central data store then sends the four sets $(P_{\mathrm{CDS}}, T_{\mathrm{CDS}}, S_{\mathrm{CDS}}, t_{\mathrm{CDS}})$ to all service providers in the set P_{CDS}, with any material authorising them to retrieve records, requesting the keys necessary for decryption. Upon receiving

the sets, the *request is checked and logged*, and the keys necessary to decrypt any records of this description are released. For all four-tuples in the Cartesian product of the sets $L_{\text{all}} \equiv P_{\text{CDS}} \times T_{\text{CDS}} \times S_{\text{CDS}} \times t_{\text{CDS}}$ the corresponding key is released, namely $\forall(P_j, T_j, S_j, t_j) \in L_{\text{all}}.K_j \equiv H(K_{P_j,t_j}, P_j, T_j, S_j, t_j)$.

The "raw" keys are released that allow the data store to detect and retrieve multiple records that fulfill the same criteria by simply looking for all IDs such that $\text{ID}_{R_j} \equiv H(\text{"ID"}, H(K_j, k))$ for $k \geq 0$. Once these records are located they can be decrypted with the corresponding key $K_{jk} = H(K_j, k)$.

The combined effect of these two protocols is that records are remotely stored, lessening the cost on the service providers and speeding the access speed for the authorities. On the other hand the administrative decisions on whether proper authority exists to access the records, as well as the logging of the criteria with which records are accessed is safeguarded. This might increase the time necessary to access records, but it is a vital safeguard against abuse.

3 Why is this not good enough?

There are several problems associated with this protocol and traffic data retention in general, that make it inherently less safe than a speedy data preservation regime, from the point of view of privacy:

- It is very difficult to ensure that data is ever deleted. Encrypted records can be stored beyond the period permitted by law, or indefinitely. More seriously decrypted records, and released key material can be saved forever. It is difficult to design a cheap system to remedy that. Note that the encryption of the records makes it easier on the side of the service provider to destroy records, by simply forgetting the session keys.
- The scheme relies on service providers maintaining few symmetric keys associated with a time period, that we have denoted K_{P_j,t_j}. This makes the scheme inherently fragile: a small leak of keys gives access to a very large volume of information. Even a temporary compromise of these keys might result in mass surveillance becoming possible [DW06].
- Since the symmetric keys have to be available on request to derive further specialised keys to decrypt records, they are exposed to abuse. Secure hardware can be used to store them and protect them, and ensure they only get released if a log entry was created. Even then the manufacturer of the hardware has to be trusted to not compromise it.
- Many jurisdiction's do not hesitate to simply seize material when it is needed in an investigation. There is nothing preventing a police force from raiding the key server of a service provider and be able to decode all previous traffic. Some jurisdiction, like the UK may ever force parties to make keys available under a gag order, which could include preventing a log entry being created.

– It is not clear how the service providers should react to authorisations that seem disproportionate (like for all users) but are none the less authorised properly. Unless an effective oversight mechanism exists this is a fundamental problem with this approach. It is not clear that such oversight regimes are possible.

In general the efficiency with which the storage and decoding process proceeds makes it all more fragile to abuse.

4 Conclusions

We presented an architecture that fulfils the stated requirements for centralised traffic data retention, namely efficiency of delivery and low cost, while at the same time limiting the potential for mass surveillance. The authorities still have to trust the service providers to truthfully package and send records, as it is the case today, but privacy is protected by a robust logging mechanisms implemented as a dual-control policy.

Even this approach has limitations, since the two parties involved in the protocols have fundamentally different powers: law enforcement comes with legal authority, massive state funding and a long history of subversion and clandestine operations, while the service providers have limited resources, are bound by commercial necessities and heavily regulated – most importantly they only have a peripheral interest in their customer's privacy.

References

[DW06] G. Danezis and B. Wittneben. The Economics of Mass Surveillance and the Questionable Value of Anonymous Communications. In *Proceedings of the 5th Workshop on The Economics of Information Security (WEIS 2006), June*, 2006.

[EPH04] A. Escudero-Pascual and I. Hosein. Questioning lawful access to traffic data. *Communications of the ACM*, 47(3):77–82, 2004.

[GSMB03] Eu-Jin Goh, Hovav Shacham, Nagendra Modadugu, and Dan Boneh. Sirius: Securing remote untrusted storage. In *NDSS*. The Internet Society, 2003.

[LKMS04] Jinyuan Li, Maxwell N. Krohn, David Mazières, and Dennis Shasha. Secure untrusted data repository (sundr). In *OSDI*, pages 121–136, 2004.

[SK99] Bruce Schneier and John Kelsey. Secure audit logs to support computer forensics. *ACM Trans. Inf. Syst. Secur.*, 2(2):159–176, 1999.

[Tay06] M. Taylor. The EU Data Retention Directive. *Computer Law & Security Report*, 22(4):309–312, 2006.