A Private and Anonymous Data Repository Service

Jarrold Trevathan and Wayne Read
eResearch Centre – James Cook University
Email: eResearch@jcu.edu.au

Abstract—This paper describes a security problem involving an online data repository which acts as a proxy for multiple companies allowing their customers to perform online services (e.g., pay invoices). The repository's host is trusted to honestly fulfil its duties in maintaining the data in a manner consistent with each companies' required services. However, the information stored by the repository remains private in that the repository's host cannot openly read any companies' operational data, nor does it learn the identities of any companies' customers. We contrast several approaches describing their viability for web deployment using existing technologies. This is a fundamentally new security problem with no established literature or clearly defined cryptographic solution. The project originated from a commercial attempt to design a secure online data archive that allows a customer to pay and view invoices online.

Keywords: Authentication, encryption, eCommerce

I. INTRODUCTION

Since the 1960s, database technology has rapidly permeated all facets of society and transformed the way in which humans store and process data. The advent of networks and e-commerce now allows database systems to be distributed and remotely accessed by users anywhere around the world. However, despite the overwhelming benefits, there are numerous security concerns that limit an online database's effectiveness. Some of the problems facing an enterprise with regard to securing its data in a networked environment include:

- 1) Access Permission – Users should only be able to access information that they are entitled to.
- 2) Denial of Service – Users performing computationally intensive queries may swamp system resources, thereby denying other users access.
- 3) Logging – User access to data and modifications must be logged to ensure possible security threats are monitored.

These concerns have been addressed in literature (e.g., see [7], [8]) and most commercial database management systems provide support for authentication, concurrency controls and recovery mechanisms. However, the solutions proposed are only for the *standard* database scenario where the repository is owned and maintained entirely by a single controlling enterprise. The enterprise has complete access to view and modify the data. However, this may not always be the case.

We propose a new system where the data repository stores data that is not its own. That is, the repository provides a service by managing data that belongs to other companies. This raises new privacy and security concerns that have not been previously covered in the literature. Specifically, the repository manager should not have the ability to freely browse archived data that belongs to someone else. All repository

users must trust that the manager will not reveal or sell potentially sensitive information to unauthorised parties.

This paper describes a security problem involving an online data repository which acts as a proxy for multiple companies allowing their customers to perform online services (e.g., pay invoices). The repository's host is trusted to honestly fulfil its duties in maintaining the data in a manner consistent with each companies' required services. However, the information stored by the repository remains private in that the repository's host cannot openly read any companies' operational data, nor does it learn the identities of any companies' customers. We contrast several approaches describing their viability for web deployment using existing technologies. *This is a fundamentally new security problem with no established literature or clearly defined cryptographic solution.* Note that this problem is different from a third party service that just merely hires out online storage. In the scenario we address, the third party is unable to access any of the content it stores, but still is able to ensure that only authorised users can access the information.

The project originated from a commercial attempt to design a secure data archive that allows a company to upload invoices from a popular accountancy program onto the data repository. A customer can log into the data repository via the Internet and can pay the invoice, which is then downloaded by the company and stored in its own personal database. This system has been deployed commercially by Osmotion Pty Ltd¹ which is a software engineering company that offers services to the Australian Mining Industry and small to medium business.

While the resulting implementation is not necessarily ideal, this paper's purpose is to highlight the specifics of this problem and the shortcomings of the existing cryptographic mechanisms that must be combined to provide the desired level of security, authentication and anonymity. We hope that this work will form the basic ground work for a solution, and also will inspire other researchers to propose more sound and rigorous mechanisms to solve the problem proposed.

This paper is organised as follows: Section II describes the problem motivation, the processes/parties involved, and the privacy and security requirements for such a system. Section III presents a protocol to register users and maintain anonymity. Section IV contrasts the differing approaches for ensuring the privacy of the database contents. Section V discusses what security mechanisms can be used to protect communications between the parties involved. Section VI provides some concluding remarks.

¹<http://www.osmotion.com.au>

II. PROBLEM MOTIVATION

This section describes the initial problem motivation, the parties and processes involved, and the privacy and anonymity requirements. It also specifically outlines the assumptions and practical constraints imposed upon the system.

A. The Data Repository Service

Basically, the *Data Repository (DR) Host* provides an online service for a company's customers. This removes the need and expense for the company to maintain its own website/online services for its customers. The costs for a business to create and maintain a website include:

- Salary for web programmers and graphic artists who must design and update the website;
- Hardware/software for developing and hosting the website;
- Ongoing costs for Internet access; and
- Administrative costs for handling the incoming/outgoing data and transactions.

A central data repository can spread these costs out amongst its clients (i.e., multiple companies using the service) to provide a less expensive service. This is especially attractive for small business where cost may be the limiting factor that prevents them from engaging in e-commerce. For example, the DR Host can offer a subscription to the system for \$100 a year, which is substantially less than the aforementioned costs.

To make this scenario more practical, the company should not be required to constantly be online by the DR Host (i.e., need not have a permanent connection with the DR Host). Instead the company can log on to the DR Host's website at will. It is the DR Host's duty to maintain a constant online presence so that the company's customers can log on to the DR Host's website and access the company's services (i.e., the system is *asynchronous*).

The repository is scalable in that the Host can offer the service to multiple companies simultaneously using the same database. That is, the DR Host can offer *separate web portals* for each company using a single server. This gives the appearance of a unique website for each company. However, for ease of discussion, we will only refer to a single company throughout the paper (i.e., the discussion will be from the viewpoint of a single company using the service).

Many security proposals require elaborate mechanisms that are clearly not practical for immediate commercial deployment. In general, it is desirable that the customer *should only be required to use a standard web browser* when accessing the DR Host's website. That is, the customer should not have to install a complicated application to interact with the DR Host's database. The DR Host must provide a quality service using only **existing web technologies**.

Note that this constraint is the single-most important factor for the proposed system. It is this factor which renders the majority of cryptographic solutions commercially unviable for the task. The reader must keep this constraint in mind when considering the discussions in the coming sections.

B. Parties and Processes Involved

This section briefly describes the parties and processes involved in the online data repository system. These procedures are elaborated upon in later sections.

There are three main parties involved:

- *Data Repository Host* – Manages an *encrypted* database of customer records and provides a web interface for accessing these records;
- *Company* – Uploads/downloads customer records to/from the DR Host; and
- *Customer* – Accesses the database via a DR Host's web interface to view/update customer files.

The proposed system consists of the following procedures and desirable characteristics:

- 1) *Registration between the company and the DR Host* – the company registers with the DR Host and specifies the type of services it requires;
- 2) *Company uploads/downloads records to/from the DR Host* – Ideally the company does not need to be permanently online. Rather the company should be free to upload/download customer records to/from the DR Host's database at any time;
- 3) *Solicitation of a company's customers* – The company solicits customers regarding the service available (i.e., advertises the service). No further direct interaction should be required;
- 4) *Registration between the customer and the DR Host* – the customer registers with the DR Host. the customer can then access the DR Host's website in the same manner as any other e-commerce application; and
- 5) *Customer uploads/downloads records to/from the DR Host* – the customer *anonymously* accesses the DR Host's services via a web interface to retrieve and modify records stored in the database.

C. Privacy Requirements

In addition to the regular database security issues outlined in Section I, the proposed scenario has many unique security and privacy concerns. This section outlines these concerns and assumptions regarding the system.

Firstly, a company (and the customer) may be uneasy in trusting the DR Host with its operational data. That is, an unethical DR Host may attempt to browse, modify or sell everything stored in its database. As a result, some preventative measures must be in place to ensure the DR Host cannot do this. However, this must be weighed against the amount of information the DR Host requires to perform its duties.

The problem lies in how to encrypt/decrypt the records in the database without the DR Host gaining access to the records. This is compounded by the fact that a customer should not have to remember a lengthy encryption/decryption key, nor should they have to use special software to interact with the DR Host (but rather use a standard web browser).

TABLE I
SECURITY AND PRIVACY REQUIREMENTS FOR THE DATA REPOSITORY
APPLICATION

Confidentiality	The data repository contents remain concealed from the DR Host and those not privy to the information
Accountability	The DR Host can verify that a customer is legitimate and has authorised access to his/her records
Anonymity	The DR Host does not learn customer identities

Fig. 1. The registration process between a company and the DR Host, and then between a customer and the DR Host

Furthermore, anonymity is an issue in that the DR Host must not keep records regarding customers' identities and which companies they belong to. Instead, customers should be able to use the service in an anonymous manner. However, the DR Host must still be able to ascertain that a customer is genuine, and only has permission to access files that s/he is entitled to.

Table I summarises the desirable security and privacy requirements for the data repository application. The underlying constraint is that these requirements must be achieved *using existing technologies* rather than theoretical constructs.

D. Assumptions

The DR Host is **semi-trusted**. This means that the DR Host is *trusted* to maintain the records in a manner consistent with the company's policy and will not tamper with the database primary keys (used for indexing the records). However, the DR Host is *not trusted* in that it is not permitted to browse the database contents.

This paper will contrast differing practical approaches for dealing with the privacy and security issues that are specific to the private data repository problem. The following sections present several solutions for providing security and privacy based on existing cryptographic/security techniques. Each approach is scrutinized in terms of how it addresses the aforementioned privacy and security concerns (see Table I), its implications for the assumption that the DR Host is semi-trusted, the additional assumptions raised by each approach employed, and its viability for commercial deployment using existing web technologies.

III. AUTHENTICATING THE COMPANY AND CUSTOMERS

This section outlines the basic registration process between the parties involved, how they can be authenticated in future dealings, and how a customer's anonymity can be preserved.

Figure 1 illustrates the registration protocol. The first procedure in the system is for the company to register with the DR Host. This can be done offline, whereby the company makes physical contact with the DR Host, provides authenticating information (e.g., taxation and account numbers), a

description of the services required, and signs a contract (*Step 1 in Figure 1*). During this process the DR Host supplies the company with a unique username and password that it can use to log onto the DR Host's site for the purpose of up/downloading customer records. The DR Host also supplies the company with some authenticating information regarding its current customers during this stage (*Step 2 in Figure 1*). This information is referred to as a *nonce* list. A nonce stands for *nonsensical data*, and is essentially a unique random string. Each customer is associated with exactly **one** nonce (no two customers have the same nonce). The DR Host controls the nonce generation process, and ensures that no two nonces are ever the same. the company is responsible for assigning each nonce to a customer and makes sure no two customer's are given the same nonce. An example of a nonce list-customer association chart prepared by a the company is as follows:

Nonce	Customer
9L28d782	Joe Banana
87h787X2	Homer Simpson
2jl487z4	Unassigned
...	...
Gd23AF4a	Unassigned

In this example there are two customers that have been solicited for the private data warehousing service. The remaining nonces are designated as *unassigned* until the company decides to solicit some more customers. If a company runs out of nonces, they can request an additional list from the DR Host.

Registering a customer with the DR Host is a more complicated procedure and is done online. Prior to registration, the company must first solicit customers announcing that the service is available. This is done through email or a physical letter sent directly to a customer (*Step 3 in Figure 1*). During the solicitation process, the customer is provided with some unique information (i.e., a nonce) that s/he can then present to the DR Host to prove that s/he is a legitimate customer. While the nonce can be delivered to a customer via email or telephone, these mediums can still be intercepted. Although it is not perfect, physical mail is a more secure option (similar to the process involved in obtaining a personal identification number for a bankcard). However, this must be weighed up against the speed and convenience of email, and the practicality of having a customer deal with a lengthy nonce.

When registering with the DR Host, a customer presents his/her nonce as evidence that s/he is a legitimate customer of a company, and also provides an email address (*Step 4 in Figure 1*). The DR Host compares the nonce against the nonce list. If it is a valid nonce (and it hasn't previously been used), the DR Host marks the nonce as active and supplies the customer with a unique username and password (*Step 5 in Figure 1*). S/he can use this information to log into the DR Host's client-side portal and access the company's records (specific to him/her). If the nonce is invalid (does not match any nonce generated by the DR Host), or has previously been registered (it is already in use, or has previously been presented to the DR Host), the customer's registration request is rejected

and s/he is informed to contact the soliciting company.

The nonse registration system not only provides authentication, but also anonymity for a customer. That is, the DR Host knows that this customer is legitimate, but does not actually know who they are. While the DR Host does learn which company the particular customer is associated with, this knowledge still does not benefit the DR Host in any way as it still does not know the customer's identity.

IV. ENCRYPTING THE DATABASE

This section describes approaches for encrypting the database contents so that it remains private. While the some of the discussion may appear to be at a high level and considered to be superfluous common knowledge, its purpose is ensure that the reader is familiar with how encryption works. The goal is to emphasise that the private data repository problem, within the constraints of the problem definition (i.e., the DR Host is semi-trusted), cannot be readily solved by using basic encryption alone if the system is to be commercially viable.

A. Encryption and the Private Data Repository

Encryption allows two parties (referred to as the *sender* and the *receiver*) to communicate such that an *eavesdropper* cannot interpret the message. This is achieved by scrambling (*encrypting*) the message prior to transmission, and then unscrambling (*decrypting*) the message upon receipt. Before engaging in communications, both the sender and receiver must agree upon how to scramble the message (called a *cipher*) and how to ensure that only they can access the data (using a *key*). The encryption's strength depends on the key size. The larger the key length, the greater the protection offered.

The problem this application faces is how to encrypt the database contents so that the DR Host (and any other unauthorised party) can not freely view the database. However, this must be done in a manner that allows the DR Host to still operate the database from an administrator's perspective. That is, the DR Host must have unfettered access to the database tables' primary keys and system catalogue in order to operate the database. Furthermore, the customers must not have to remember/store a lengthy key, nor should they have to interact with the company once registration has been completed.

B. Externally Imposed Security Limitations

There exist government restrictions on the use of cryptography which have practical limitations for the private data repository application (see [4]). It is argued that in the interests of national security, private individuals should not be able to encrypt their communications such that a government intelligence agency cannot interpret the message if deemed necessary. Instead, commercial applications are required to use *weak* encryption. This provides a given level of privacy against general security threats, but can be eventually broken by a government agency with relative ease.

An alternative to weak encryption is *key escrow* (see [5]). With an escrow system, a third party (referred to as an *escrow agency*), possesses some information that allows it to decrypt

the encrypted information if necessary. The information held by the agency can either be a copy of the key itself, or some information that enables it to deduce the key. The agency will only decrypt the information when presented with the equivalent of a court order.

C. Private Information Retrieval

A Private Information Retrieval (PIR) protocol allows a user to retrieve an item from a server in possession of a database without revealing which item s/he is retrieving (see [1]). One trivial, but very inefficient way to achieve PIR is for the server to send an encrypted copy of the database to the user. This protocol gives the user information theoretic privacy for his/her query. Other approaches include making the server computationally bounded, or to assume that there are multiple non-cooperating servers, each having a copy of the database.

PIR is not applicable to a private data repository application. This is because the DR Host still has access to the entire repository contents even though they do not know which particular piece of information was retrieved during a query. A private data repository application requires that the DR Host does not know the repository contents **and** does not know what information the user has queried.

D. Key Exchange and Storage

Key exchange and storage are among the most vexing problems for the proposed data repository system.

Private key encryption involves using a single key to encrypt and decrypt the data. An eavesdropper who does not know the key cannot interpret the data. However, the problem with this approach lies in key exchange. That is, the sender and the receiver must exchange a secret key prior to communicating.

In terms of the private data repository application, the company and the customer must exchange a secret key prior to placing records in the data repository to ensure the DR Host and others can not read the database contents. Key exchange cannot simply be performed by email as the key can be intercepted. If the DR Host conducts this attack, it would enable it access to all the stored data for the customer.

There are elaborate protocols for exchanging keys between two parties, but the general requirement is that both parties are online. This *conflicts with the requirements* outlined in Section II that the company does not need to be permanently online either during registration, or throughout the general system operation.

An alternative approach is *public key encryption* which uses separate keys for encryption and decryption [2]. This avoids the key exchange problems associated with private key encryption. The receiver publishes his/her public key (i.e., it is accessible to anyone) and keeps his/her private key secret. The sender encrypts the message using the receiver's public key. Upon receipt, the receiver decrypts the message using his/her private key. Public key encryption requires an additional party (referred to as the *Key Distribution Centre (KDC)*) to maintain the receiver's public key (i.e., to solve the key exchange issue).

Fig. 2. Securing the database contents using End-To-End Encryption

In terms of the private data repository problem, both the company and the client would have to publish public keys on the KDC. When either party desires to upload a record, they obtain the public key from the KDC for encryption. However, this approach is rather cumbersome and requires everyone to trust the KDC (essentially trust is just being moved to another party). Although SSL uses public key cryptography, the certificate authorities are already established. It would be *expensive and impractical* to maintain KDC services specifically for the private data repository application. Furthermore, public key encryption is also *extremely slow* (compared to private key encryption), and requires large key sizes. Due to these factors, public key encryption cannot be practically implemented for the private data repository scenario.

E. Encryption Function Placement

Assuming that key exchange has already occurred, there are two places where encryption can be performed:

End-to-End Encryption. The company encrypts the data and stores it in the DR Host's database. the customer then retrieves the data from the DR Host's database and decrypts it. In the reverse process, the customer modifies the data, encrypts it and then stores it back in the DR Host's database. the company then retrieves the information, decrypts it, and updates its own database. Figure 2 illustrates this process by indicating the placement of the encryption function.

While this approach is highly secure, it does have some practical limitations when being implemented with *existing web technologies*. The main issue is that the approach requires special purpose software to be installed on the customer's machine. That is, dedicated software to perform encryption and decryption. This cannot be achieved with a standard web browser, which means that the application now becomes platform dependent (i.e., separate applications are required for Windows and Mac). Furthermore, there are some security issues regarding how to store the key. Key sizes are required to be large (up to 1024 bits). It is not practical to force a human to remember this. The key cannot be stored in a cookie, nor on the user's hard drive anywhere. During solicitation, it is unreasonable to print the key in the solicitation letter and expect the user to type it into the encryption/decryption program every time they want to use the service.

A Java Applet could be used to go some part of the way in resolving this issue. However, once again, this requires the user to download the Applet first prior to using the service. This is likely to deter users with slower connections. The problems

Fig. 3. Securing the database contents using Application Level Encryption by the DR Host

regarding key storage still remain. Furthermore, there are still some problems with platform dependency, and users must ensure that their browsers are Java compatible.

The concluding sentiment for end-to-end encryption using existing web technologies is that while it is highly secure, it is also restrictive and expensive to develop (in terms of writing applications for each platform). This seriously impedes on the system's practicality and commercial viability.

Application Encryption/Decryption on the DR Host. This approach allows the DR Host to perform the encryption and decryption. While the database remains encrypted, decryption is performed by the DR Host's software before transmission to the customer (see Figure 3). The data stored in the repository still remains secret provided the DR Host does not tamper with the application software. This would be backed by a *written privacy agreement* between the DR Host and the company stipulating that the DR Host would not act in any dishonest manner. This setup still prevents any external attack (that does not pass through the DR Host's application software) from accessing the database contents. However, the problem with this approach is that it erodes from the claims that the repository is truly private. That is, even more trust is now being placed in the DR Host.

The concluding sentiment for application encryption/decryption using existing web technologies is that it is easy and inexpensive to develop. It also avoids the problem of key exchange. However, it decreases the level of confidentiality offered by the data repository service. Although, when coupled with the nonse system, this approach can achieve *confidentiality through anonymity*. That is, even if the DR Host breaks into the application level encryption they still will not know the customer identities (as the customer's personal details are not stored in the data repository). Only the soliciting company knows the mapping between the customers' nonses and identities.

V. ENCRYPTING/AUTHENTICATING THE COMMUNICATIONS

This section outlines the threats that may be encountered during the transmission of information between the parties in the private data repository application.

A. Transmitting Information

Information transmitted between the company and DR Host, and between the DR Host and Customer is vulnerable. This can be observed by an outside party who could merely be

Fig. 4. System operation indicating the placement of SSL encrypted sessions for secure communication between parties. Unencrypted communications in these areas are vulnerable to eavesdropping, modification, fabrication and interception (blocking)

eavesdropping on messages, or who has more sinister intentions such as modifying, fabricating, or blocking the messages. This is especially a problem when sensitive data such as credit card numbers and passwords are being transmitted.

In this application, Secure Sockets Layer (SSL) can be used to establish a secured session to facilitate sending messages. The SSL protocol uses both public-key and private-key techniques to securely transfer information. Figure 4 illustrates the vulnerable communications between all parties in the system, and the placement of encrypted SSL sessions to prevent eavesdropping and other security threats.

B. Email Security Issues

Any email sent by the DR Host to the company or customer is not protected by the SSL session. As such, an eavesdropper can still read any passwords emailed. A customer obtains a password during the registration procedure. It is also possible for a customer to update a password at his/her discretion. These two procedures are relatively safe as the password can be communicated in real time under the protection of the SSL session. However, problems arise in the situation where a customer has forgotten his/her password. The typical remedy is for the customer to enter his/her email address and then the DR Host will email the corresponding password to that address. This has obvious problems as the customer's password is now exposed to anyone observing network traffic.

Online banking websites commonly have a telephone number, which a customer can call in this situation the customer would then provide some identifying information (e.g., name, address, date of birth, etc.). If the information is correct, then the bank would inform the customer (over the phone) what the password is. However, this remedy would not work with the private data repository application as the DR Host does not know the customer's identity. All the DR Host knows is the customer's nonse and email address. Password notification could be achieved by asking the customer for his/her nonse, but it is probable that the customer may not remember or have kept records regarding his/her nonse.

An alternative is to use Privacy Enhanced Mail (PEM). PEM allows mail to be signed and encrypted using certificates similar to SSL. PEM is based on a concept called *Web of Trust* in which friends authenticate each other. Provided that the recipient knows and trusts someone who has authenticated the sender, then they have some degree of confidence in the sender. Keys are exchanged by the recipient sending an email to the encrypting party which contains the key. However, PEM is not widely used due to various difficulties involving public

key infrastructure and problems with MIME encoding. Also it is more suited to personal email, not e-commerce applications.

VI. CONCLUSIONS

This paper describes an online data repository, which acts as a proxy for multiple companies allowing their customers to perform online services (e.g., pay invoices). The system spreads costs out among multiple companies and provides web hosting, thereby decreasing the normal costs associated with developing and maintaining an individual e-commerce application. The DR Host is trusted to honestly fulfil its duties in maintaining the data in a manner consistent with each companies' required services. However, the information stored by the repository remains private in that the DR Host cannot openly read any companies' operational data. This is achieved by encrypting the database's contents (although some serious practical limitations have been identified). The DR Host also does not learn the identities of any companies' customers. Each customer is issued with a unique nonse that is used for identifying purposes. The DR Host does not know the association between the client's identity and the nonse they are using. This paper contrasts several approaches describing their viability for web deployment using existing technologies.

This paper has presented a fundamentally **new cryptographic problem** with no existing solution. While some of the discussion may seem to be at a high level, this is only for the purpose of showing what security building blocks can be used to help solve the problem, and the apparent strengths and weaknesses of each approach employed. While it may appear that this is a straight forward encryption problem, further analysis of the requirements and the constraints of using existing technologies during the implementation severely compounds things. No single existing security mechanism appears to be adequate. Instead the private data archival service requires several mechanisms to be applied in unison. The selection of certain approaches has definite trade-offs and undesirable consequences for other aspects of the system. It is left as an open problem for the cryptographic/security community to propose a more rigorous and sound solution.

REFERENCES

- [1] Chor, B., Kushilevitz, E., Goldreich, O. and Sudan, M. (1998), Private Information Retrieval, *J. ACM* 45(6), 965-981.
- [2] Diffie, W. and Hellman, M. E., (1976), New Directions in Cryptography, *IEEE Transactions on Information Theory*, vol. IT-22, 644-654.
- [3] Inmon, W. and Hackathorn, R. (1994), *Using the Data Warehouse*. John Wiley & Son's, ISBN 0-471-05966-8.
- [4] McCullagh, D. (2001), Congress Mulls Stiff Crypto Laws. *Wired*. <http://www.wired.com/politics/law/news/2001/09/46816>
- [5] Micali, S. (1993), Fair Public-Key Cryptosystems. *Advances in Cryptology - CRYPTO '92 Proceedings*, Springer-Verlag, pp. 113-138.
- [6] NIST, Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, Special Publication 800-67.
- [7] Rosenthal, A. and Sciore, E. (2000), View Security as the Basis for Data Warehouse Security. In *CAiSE Workshop on Design and Management of Data Warehouses, Stockholm*.
- [8] Warigon, S. (1997), Data Warehouse Control and Security. *Association of College and University Auditors LEDGER*, Vol. 41, No. 2, April 1997, pp 3-7.
- [9] Widom, J. (1995), Research Problems in Data Warehousing. In *Conference on Information and Knowledge Management (CIKM)*, pp 24-30.