Translucent Databases:

A Precursor to Privacy Sensitive Databases

Palaka Bhattacharya
Morteza Karyar
CSCI E170- Security, Privacy, and Usability, Fall 2004

Table of Contents

1. Abstract

A translucent database, as Peter Wayner defines it, is a database that "lets some light escape the system while still providing a layer of secrecy." [1] A privacy-sensitive database is a database that stores personally identifiable information and guarantees the security of such information. Currently, most databases do not provide security for personal information. They tend to store all kinds of personally identifiable information regardless of whether the gathered information is actually necessary. What is worse is that normally no security measures are taken to keep such data secure. Many database designers do not perceive the social dangers of storing so much personal data but rather, believe that even if all the data is not being used, it will come of use someday. Such pervasive but unnecessary data collection, accompanied with the explosion of available storage space, and un-sanitized discarded storage, poses a great threat to the basic right of people to privacy. We examine in this paper some of the ideas developed by Peter Wayner in his book *Translucent Databases* and analyze their relevance to privacy-sensitive databases. Building on the concepts proposed by Wayner one can produce privacy-sensitive databases. We briefly discuss issues regarding usability and security of these databases and some of the requirements for their spread. These requirements are 1) an active campaign for legislating laws that demand enforcing people's right to privacy and 2) a conscious effort by all involved both in the business community and in the software industry to promote the use of privacy-sensitive databases.

2. From Translucent To Privacy-Sensitive Databases

Wayner, the author of *Translucent Databases* and the person who first coined this term considers these databases as "a compromise between complete transparency and complete opacity."[2] Admitting that the term is rather "a vision and a goal than a well-defined predicate," he notes that translucent databases let necessary information out while providing a layer of security.

Wayner agues that information that resides in the clear and behind the walls of security mechanisms is still vulnerable not only to outsiders who can penetrate the walls but also to insiders within the walls. Many instances of information disclosure through insiders confirm the severity of the problem. Wayner maintains that translucent databases are a reliable solution.

The basic idea is that precisely that information that should not be seen should be hidden, either by encryption or by other similar methods. The rest of the information can be kept in the clear. While encryption is not a new idea, the originality of Wayner's thought lies in his interest in keeping the prying eyes of the insiders, the database administrators, the clerks, the curious, as well as hackers and the like away from people's private information.

[1] Peter Wayner, *Translucent Databases*, (Flyzone Publishing, Natick, MA, 2002), Page 2
[2] ibid, page 9

Some of Wayner's examples are more in favor of protecting privacy, while others are strictly security related without a necessity for privacy. One should note that, while protecting privacy normally involves some security measures, security is not always used to protect privacy. Protecting the privacy of client information in a patient's database requires security, whereas providing security for a missile launching system does not necessarily have anything to do with protection of privacy.

Wayner's initial indication of what translucent databases would achieve are promising. However, his original description of what these databases are is faded in his extensive effort to explain a diverse range of cryptographic algorithms and to illustrate how they can be used to implement secure databases. He explains and implements many cryptographic algorithms such as hash functions, chained hash functions, symmetric encryption, public key cryptography with a single private key and with multiple private keys, bit commitment, digital signatures, and more. He demonstrates the use of salt, sequence numbers, random numbers, challenge numbers, and a number of information hiding techniques such as quantization and usage of fake data.

This extensive coverage of cryptography distracts Wayner and while trying to illustrate translucent databases, he provides algorithms for databases that are not translucent but completely transparent or completely opaque. This leads the reader to a confused understanding of the nature of translucent databases. Looking at Wayner's examples one can recognize three categories:

a. **Translucent databases**: These include most of the examples. A partial list includes databases for a babysitter system, a rape victims' system, a cancer research system, a complaint box system, a committee suggestion box system, a department store ordering system, a rental car system, a payroll system, a prescription system, a logfile, an appointments system, and a ship locator system.
b. **Opaque databases**: The example here is a missile launch system.[3]
c. **Transparent databases** that are accessed by secure algorithms: the example here is a stock quoting system.[4]

A closer look at all Wayner's examples shows that translucent databases are meaningful often when they are used for protection of privacy. Except the ship locator system most of the other examples are either already designed to protect privacy or a slight change in them would make them so. The correlation between translucent databases and databases that protect privacy is not accidental. Wayner, in his many examples throughout the book, shows great interest in keeping private information secure.
We should note here that, as mentioned before, translucent databases also include some databases, such as the ship locator database, that do not hold privacy-related content. Furthermore, we should note that, although translucent databases are mostly privacy-related databases, they do not cover all such databases. In some cases a database that protects privacy may require complete opacity.

[3] ibid, pages 83-84
[4] ibid, pages 171-172

This brings us to the realization that using cryptographic algorithms and benefiting from the methods that Wayner provides we can take one more step and define a class of databases that are specifically designed to guarantee protection of privacy. Such a class could be called privacy-sensitive databases which is what we are interested in. Proposing such a class is not redundant since it narrows down translucent databases to only those that hold and protect personally identifiable information, and it expands them to include those opaque databases that enforce privacy protection. For the rest of this paper, while looking at Wayner's translucent databases our inclination will be to further define and advocate privacy-sensitive databases.

2.1. Privacy-Sensitive Databases And Usability

Any implementation of a privacy-sensitive database must consider the usability problems created by one-way hash functions, which we will for brevity call hash functions. Since hash functions are used to scramble data such as names of people, retrieval of information on a particular data item needs the exact spelling as the original one. Any change in the entry, when hashed, creates a completely different hash value. For instance, the hash of 'Fred Smith' is going to be completely different from that of 'Fred Smith' with one additional space and that of 'fred smith' without capitalization. The reason is that good hash functions scramble similar data into completely dissimilar hash values. Following are the values generated from the above names using md5 hash function:

md5('Fred Smith') = e087aa7eade9b41ebd0063e64ec5c89c
md5('Fred Smith') = 2a0cbb5625ff80fe45a7a83d7d09a2d4
md5('fred smith') = a7ad364dccd4266d0940a650ea3c48eb

The usability problem arises when ordinary people are exposed to the unforgiving nature of hash functions. It is quite natural for people to talk about the same thing in many different ways. Similarly while doing data entry into a database, people may capitalize a name or use its short form or spell it in different ways. The hash of a name or any string can only be recognized a second time if it is spelled exactly the same as the first time. Search of similar items yields no results. In such cases, browsing the database in search of the string is not useful either, because the string is saved in a hashed format and is not recognizable. This is especially important since forgetting the exact spelling of a stored item may lead to its permanent loss.

A quick look at Wayner's views of usability is useful since other system designers may share them. Wayner recognizes the usability problem and provides a number of solutions.[5] However, he does not seem to take this problem seriously enough. When talking about simple solutions, Wayner describes the necessity for an exact input as a painful training that "exposes the inner pickiness of computers in a way that can be useful to humans capable of adaptation."[6] He does not realize that such a system suffers so severely from lack of usability that it would be abandoned before becoming useful.

To successfully promote privacy-sensitive databases, we must pay very close attention to usability issues. There is no reason that people would want to continue using a system if

[5] ibid, page 31
[6] ibid, page 30

it suffers from a severe lack of usability. The belief that the strictness of hash functions can be useful is a harmful one and must be abandoned.

We do not intend to provide solutions to such problems in this paper but one simple solution that may work in some cases comes to mind. One can keep an auxiliary table that holds the unique spelling of a number of random names in addition to names that are hashed in the database. A random browser does not get any information by looking at this table because it does not contain anything but a list of random names. On the other hand, when the search of legitimate users does not yield results, the software can use the auxiliary table to find similar spellings and look for them. Other solutions must be devised to make privacy-sensitive database acceptable.

2.2. Wayner's ideas Suitable for Designing Secure Privacy-Sensitive Databases

Wayner has many interesting ideas that can be used to design privacy-sensitive databases. These ideas are especially important since they secure data even against insiders.
The basic tool that Wayner uses is simple one-way hash functions. Hash functions such as MD5 and SHA-I are available for many languages and are considered secure. A simple hash of a name denoted as h(name) creates its scrambled form, which hides the actual name when uninvited browsers look at the database. A more secure option would be to add a password to the hash as in h(name, password) or even add additional items such as a serial number as in h(name, password, serialNumber) or a challenge number or a combination of them. These additional ingredients make the unauthorized retrieval virtually impossible.
We will look at some of the other cryptographic tools that Wayner uses when looking at some of his examples. First we will try to classify these examples. Privacy-sensitive databases that Wayner examines can be grouped into two classes:

a. Those that deal with handling information that almost everybody would agree must be kept secure due to privacy reasons. They include: a babysitter database, a rape victims' database, a cancer research database, a complaint box database, a committee suggestion box database, and a prescriptions database among others.
b. Those that are associated with databases whose owners would need some additional incentive to make the effort to secure the database. They include systems such as: department store ordering, rental car, payroll, logfile, benefit notices, and the like.

Promoters of privacy-sensitive databases can begin with the first group and as their advantages and simplicity becomes more apparent to the public, and hopefully better privacy laws are put in place, proceed to the more difficult group. We, too, will start our discussion with the first group.

2.2.1. Privacy-Sensitive Databases That Are Easy To Promote:

Wayner provides some good examples for privacy-sensitive databases that are easy to promote and implement. We will briefly look at a few of them that show how easily a privacy-sensitive database can be built. They include: a cancer research database, a rape

victims database, a complaint box database, and a suggestion box that can only be opened by all members of a group, among others. The babysitter system, a famous example by Wayner, will be examined separately. The first four examples are great cases to use to promote simple and inexpensive privacy-sensitive databases. The first one uses hash functions, the second adds noise before hashing, and the last two use two varieties of public key cryptography.

2.2.1.1. Use of Hash Functions in a Cancer Research Database:[7]

Most people would not be comfortable with being part of a research program if their personal information were not secure. The cancer research table is a very good example that shows how one can design a privacy-sensitive database without expensive secure servers. In this example, a table keeps research information on correlation of cancer with people's smoking habits, amount of radon in their environment, their residence, and other related data items. While a standard table would keep everything in the open, in Wayner's solution one can keep the person's name and if necessary the street name or number or both in a hashed form. A browser won't be able to see a list of names with their data while researchers can add new data on each person or collect aggregate data on the basis of place of residence or other data items. Following is a sample table entry for two people with the names Peter and Paul where the hash function is MD5

NameID	StNum	Street	Zip	Radon	Smoke	Cancer
6fa95b1427af77b3d769ae9cb853382f	5678	8355c2…	02145	1	6	1
c13e13da2073260c2194c15d782e86a9	1234	641ee1…	02145	6	6	8

Serious intruders would still be able to decrypt information if they knew the name of a subject of research. An easy solution to this latter problem would be adding a password to the hash as in h(name, password). Now only people who know both the name and the password can retrieve decrypted information. Following is sample table entries for the same people as above but with the password '2BeSecret'.

NameID	StNum	Street	Zip	Radon	Smoke	Cancer
fc60873c62600515c6d4cd7297fb586f	5678	8355c2…	02145	1	6	1
d436d1062de292c0104e88403510ece9	1234	641ee1…	02145	6	6	8

Since such a program would normally be used by a limited number of researchers, a single password would suffice. Other more complex solutions can and must be used in more serious situations.

2.2.1.2. Use of Random Noise in a Rape Victims System:[8]

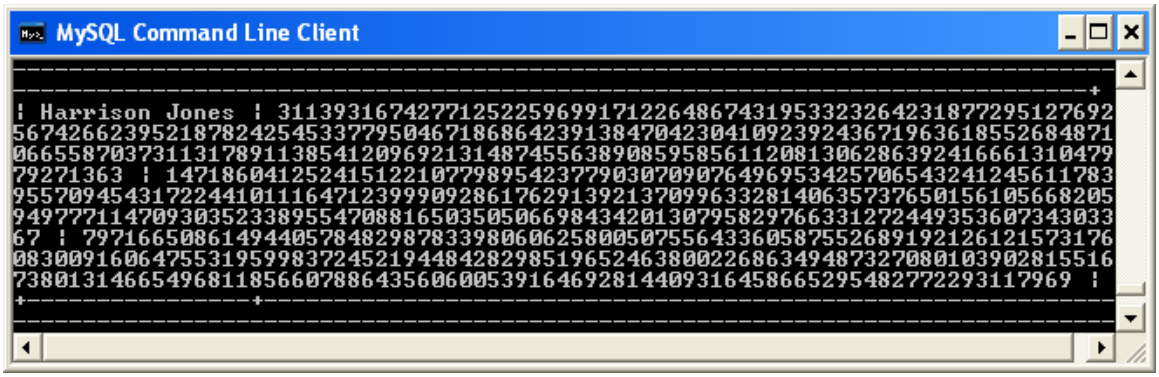
This is another system, which is easy to both justify and implement. This system uses the same principles as the cancer research does in addition to some random noise to permanently obscure the name of the victim. The random noise can be generated by adding a random number for each day. Keeping the same random number for each day is

[7] ibid, page 124
[8] ibid, page 126

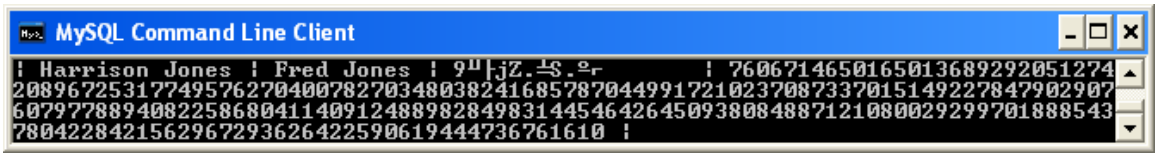
necessary for linking to data in other tables during the same day. Such a database can then be used openly for research purposes without any privacy issues. This would limit the usage of expensive and complex systems only to where permanent preservation of all data is necessary.

2.2.1.3. Use of Public Key Cryptography (PK) in a Complaint Box System:[9]

Wayner's complaint box system uses two tables: one for public key ingredients and the other for messages. The public key table holds public key information such as name, g, g^x, and p. A public key entry for a person by the name of 'Harrison Jones' is shown below in MySQL format. Here, the fields are separated by a vertical bar '|':

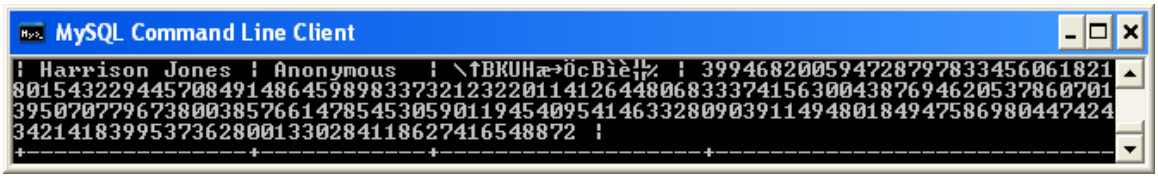


The long numbers represent the values of g, g^x, and p.
The messages table in Wayner's example holds the name of the sender, the name of the receiver the encrypted message and g^y. To make the system privacy-sensitive one should eliminate the senders name altogether and add it to the message which is encrypted. That way intruders would not know who is sending the message. To send a message, the sender uses the public key to encrypt the message and puts it in the messages table. An entry in the message table sent from a sender 'Fred Jones' to a 'Harrison Jones' containing the message "Hello Harry" is shown below:



Here, the message is encoded using mySQL's Encode function and the public key. The long integer is the value of g^y. The more privacy-sensitive message 'Fred: Hello Harry" sent as an anonymous sender has the following entry:

[9] ibid, page 55



To open the message the receiver decrypts the message with the private key. This system uses a translucent database and very simple cryptography to facilitate a secure and privacy-sensitive in-house complaint box. Here, similar to the previous systems no expensive secure server is needed. To provide usability, designers should hide the inherent complexity of the system from end users.

2.2.1.4. Use of PK with Multiple Private keys in a Committee Suggestion Box System:[10]

The committee suggestion box system is very similar to the previous example but here a group of people must all agree to decrypt a message. This example uses the simple xor bit operation denoted as '⊕' to distribute one secret key among multiple people. Given n committee members, n-1 random numbers with the same number of bits are chosen. Then the nth number is chosen such that when it is xored with the given n-1 numbers it produces the private key.

$x_n = x_1 \oplus x_2 \oplus \ldots\ldots\ldots \oplus$ secret key

Each member of the committee is given one of the n numbers. Encoding and sending of a message, which includes the name of the sender, can be done by anyone using a general public key that is in the clear. For decoding, Wayner provides a simple procedure, which allows opening of a message by collective work of the group without disclosure of any of their private keys to each other. In this example, Wayner also uses digital signature concepts to provide for the integrity of the message.[11]

As these examples show privacy-sensitive databases can be implemented easily and without expensive secure servers. The examples are all cases that can be used to promote privacy-sensitive databases and serve as models for many other situations.

2.2.2. Privacy-Sensitive Databases That Need Additional Effort To Promote:

Unlike the previous group of examples, not all privacy-sensitive databases are easy to promote. One of many examples from Wayner's book that is in this category is a department store ordering system.

Here, one option for a privacy-sensitive database is, as Wayner suggests, using a translucent database that does not give the database owner access to customer names. In Wayner's opinion this is not a problem since he believes that "The information is […] not crucial to the store business," and that, "There is no solid reason why a store needs to

[10] ibid, page 88

[11] A simple version of all the above examples is implemented by Wayner and the code is generously provided on the book's website. However, the code does not work out of the box and some tweaking is necessary.

know who bought what when."[12] Wayner seems not to be aware of the current market for personal information. Businesses do use their customer's personally identifiable information for marketing and would at times purchase such information. Therefore, the simple version of the ordering system that takes database owner's access away would not be acceptable to many owners.
Promoting a level of privacy protection that is still not acceptable may have a negative effect and result in no protection at all. Databases do not need to take all access away to become privacy-sensitive. There is a continuum and a system designer should try to achieve as much protection of privacy as possible. One solution for the ordering system is to design a system that instead of customer names keeps only their hash at the clerks' desks. This would take a major source of abuse away and is the solution Wayner proposes for a prescription drugs system. Securing personally identifiable information at the main servers of the department store is still possible. There, one can still use more complex designs to keep the private data encrypted while giving access to the owner at the same time.
The main point is that even without privacy laws that would require serious protection of customer information, system designers can still attempt to promote privacy-sensitive databases in an acceptable form. Asking for too much may yield no results.

2.2.3. A Critical Review Of The Babysitter System

An important factor in building privacy-sensitive databases is to guarantee that their private information is secure. This point might seem obvious but a close look at the babysitter example[13] in *translucent databases* shows that cryptographic algorithms do not necessarily provide security.

The security of the babysitter system is of especial importance, since as Wayner notes, its database "contains the whereabouts of minors," and information on houses without parents.[14] This example promotes the kind of design that is clearly vulnerable and carries the seeds of its own failure. In order to promote and spread privacy-sensitive databases one needs to start from secure and usable examples.

An important point here is that the babysitter example with all its security problems is given by Wayner whose expertise in cryptography is obvious. Wayner, like many other experts on cryptography, cares about privacy. However, his attention is distracted by attempting to illustrate as many cryptographic algorithms as possible. The care for cryptography overrides the care for security and protection of privacy and this is what designers of privacy-sensitive databases should resist.

Wayner dedicates a whole chapter to this program and produces several solution versions. He plants the seed of the problem when he declares that since "babysitting is not a lucrative job, so baby sitters are unlikely to pay much for the scheduling service."[15]

[12] ibid, page 28
[13] ibid, page 57
[14] ibid
[15] ibid

Therefore, Wayner suggests that storing the data in a cheap database using cryptographic techniques that are implemented in "a few minutes" is the solution.

The babysitter program keeps information on teenage babysitters and parents and links the two groups while protecting the information from others. The first version that Wayner offers hashes each babysitter's name and password as in h(name, password) and saves that as the identifier for the sitter. Anyone who needs to access the sitters' schedule must have their names and passwords. The name of babysitters is not available to people who browse the database. Only people who already know the names of the sitters and their passwords have access.

But, the problem in this program is that everyone who may need a babysitter must have all sitters' names and passwords. Moreover, these passwords are unchangeable since changing them makes all previously entered information invalid. Clearly, a permanent password that is disclosed to others cannot be a serious solution. This is against any basic security standard. Moreover, this is bad security training. However, Wayner's next version that attempts to improve this solution does not address this security issue but the problem that "the sitter can't lock out individuals,"[16] with whom her relationship goes sour.

This improved version hashes the name of the sitter twice, each time with a different password as in h(name, password1) and h(name, password2), and each password is given to one group of parents. Now, if the relationship of the sitter goes sour with one parent the sitter can block the group that has the same password by not entering any more information with that password. One would normally think that such a program that denies access to a whole group to deny access to one person is unacceptable. But Wayner doesn't seem to be bothered by this problem and uses this version as an improved example.

Wayner's third solution has a parent's table to avoid the group-denial problem and has a sitter's password only known to the sitter. The sitter password is a definite improvement. When the sitter wants to authorize a parent to look at her schedule, she adds the hash of their name and an optional password they might have to a record that contains the hash of her own name and password. The hashes would look like the following:
hash(parentsName, optionalPassword), hash(sitterName, sitterPassword) .
Wayner suggests that a single optional password be used for all parents. Now the parents can be dropped from the list individually and the sitter's password is not known.

This solution still has the great advantage of anonymity of the people involved but other problems persist. Since parents don't have the sitters' passwords, if there is more than one sitter in the system, they cannot tell who is who. This problem makes the solution practically unusable. Moreover, anyone who baby-sits once will learn the parents' password (or any specific parent password if one available) and has the capability of misrepresenting them or distribute all such passwords to others. Wayner has given three

[16] ibid, page 59

versions of the implementation and has yet to dealt with this basic security issue. Moreover, he does not clarify why such relax security is allowed.

The fourth solution attempts to solve the babysitters' anonymity but keeps the previous problems and creates new ones. As before, any sitter who baby-sits once or even applies for babysitting needs to know the parents' password, which due to the nature of hash functions is a permanent one. Moreover, anyone who has the parents' password, which is the same for everyone, can see all babysitter schedules. What is added in this final version is that the names of the sitters are also in the open.

While Wayner considers the use of distinct passwords for each pair of parents at one point, he recommends one password for all parents. Within his design this make sense since otherwise, each sitter would have to learn every parents' password. The requirement that the sitter must know the parents' password(or passwords if distinct passwords are assigned) is a loophole that, as the password(s) spreads, opens the security hole wider and wider.

Wayner's solutions to the babysitter problem, whether the earlier ones that are quick and simple or the ones that are more complex, are all toy solutions and cannot be considered for a serious and secure database that involves more than very few people. These solutions provide bad examples both for promoting security in general and for promoting privacy-sensitive databases.

3. Implementation Considerations of Privacy-Sensitive Databases

Sensitive data is an obvious target for attackers. While Wayner's book discusses several techniques for ensuring the internal security of data while it is at rest, it is just as important not to send private data in the clear across networks. The overall security infrastructure of an organization must not be overlooked. The data needs to be encrypted at all times.

Encryption must also happen at the storage level. This is another layer that is overlooked. One must ensure that the data is encrypted at the point when the data is written to disk. Encrypted fields take up more storage space so database designers need to allocate sufficient disk space. Field types need to be adjusted as well, for instance, numeric fields should be changed to character fields that are large enough to store hashed data. [8]

Organizations implementing encrypted databases need to thoroughly plan out a security model that will ensure that keys are not readily available to a lot of people. One needs to ensure that the decryption key is stored away safely and not transferred over the network. The decryption key can be stored using many methods [9]:

- The key can be stored encrypted in a table within the database. Oracle databases use this method. The stored keys themselves are decrypted and therefore not readily visible to even the database administrator to use. Only users with access rights can only decrypt the keys and then decrypt the desired data.

- One can also use third party software like RSA Security or nCipher hardware where the data gets encrypted when being entered into the database and decrypted when queried.
- One can also store the key on a disk and ensure that the disk does not reach people who should not have access to it.

Last, but not the least, organizations should also consider the overhead cost of implementing a database where there are fields that are encrypted. Of course, the more fields that are setup for encryption, the more overhead there will be in encrypting or decrypting and then writing to disk and in performing disaster recovery backup jobs. Disk capacity that will be needed will increase, thus increasing the cost of the solution. This solution also takes a hit on application performance. Troubleshooting corrupted queries becomes trickier and may end up taking longer, and in a fast paced industry like the stock market, the luxury of time is not always available.

However, at the end of the day there is nothing more important than knowing that an intruder is kept out of a system that stores personal customer data.

4. What we can do to raise the awareness of privacy sensitive databases

The compelling need for organizations to adopt privacy sensitive databases must stem from stronger privacy laws that enforce and encourage preserving anonymity in databases. While some of Wayner's ideas for preserving personal information in databases are novel, they are rarely being used commercially today as there is no primary driving force pushing for the privacy of personal data. In order for more rigorous data protection to take place, the following must occur:

- There needs to be a more aggressive campaign for privacy rights within the United States
- The level of awareness of system designers towards preserving personal data internally within databases needs to be heightened
- Database owners need to be increasingly made aware that this function is primarily their responsibility
- The privacy laws in the United States towards preserving anonymity in the commercial realm could be extended by using the privacy laws of the European Union (EU) and Canada as a model.

4.1. Active campaign for privacy rights

Personal information collection is extremely pervasive in our current society. Albrecht and Ed. M. report in "Supermarket Cards: The Tip of the Retail Surveillance Iceberg" [10] that there exists en enormous "data warehouse" derived out of consumer information that is filled in when customers apply for discount retail shopping cards. This information often includes details such as a driver's license number. This number in a lot of cases is the same number as their social security number. Consumers often believe that providing their private information is necessary to reap discounts and countless other retail benefits.

Few realize that the possibilities for the abuse of such sensitive information are endless. Consumer purchase patterns together with personal details such as names and addresses when stored in clear text in retail databases, can be aggregated and sold for instance to credit bureaus, marketers and other retailers and even to health insurance companies who could potentially base consumer rates on spending patterns.

Since few people realize the dangers of such ubiquitous data collection, pro-privacy advocates need to conduct a more active campaign for privacy rights that educate people on the dangers of personal information being readily collected and possibly abused in systems. People need to understand and be forewarned of the objectives of systems that collect their personal data and the limits to which such data will be used. With a more focus on privacy rights in our society at the grassroots level, stronger privacy laws that push for privacy sensitive databases can be put into place.

Werner's book, "Translucent Databases" is an excellent propaganda tool for pro-privacy advocates because scenarios in the book where the dangers of having personal details in the clear such as in the babysitter's database or the details of rape victims being transparent in a police database are likely to raise great public empathy for the need to protect personal data. Cases such as these should be brought to the forefront by privacy advocates in the media as should actual criminal cases of privacy abuse.

4.2. Raising designer consciousness

Just as an aggressive campaign for privacy rights needs to be put into place, database designers and programmers need to be conscious of the social ramifications of designing systems that collect personal data in order to make privacy sensitive databases commonplace.

Software companies, particularly those that are industry leaders, should play an active role in making the software design community aware of privacy issues. Database designers should include a formal assessment of the nature of data being collected on people and the need for their collection as part of the software design/engineering lifecycle. Audits should be conducted by both external and internal security experts on all legacy and current systems at least once a year to determine if privacy breaches are taking place.

The issue of databases that do not preserve anonymity will be less prevalent if system designers and programmers realize the need for anonymity in our society today and work pro-actively towards building trusted systems.

4.3. Database owners and their responsibilities

Database owners need to recognize that it is essential that they collect only as much personal data as is necessary to accomplish the tasks at hand. Justification must be given for whatever personal data that is collected and people should always be given the option to opt out of providing their personal information where collection of it is not mandatory

to the tasks at hand. In addition, database owners within a corporation must ensure that strong internal corporate policies are adapted that strictly prohibit and punish data misuse from insiders. Database owners also need to recognize that they too could be held potentially liable for any damages caused by the misuse or wrongful exposure of personal data from their databases.

4.4. Extend US Privacy Laws

Companies operating in the United States aren't required by law to offer any specific privacy protections unless they're in a regulated industry, such as collection agencies, financial services and health. The Fair Credit Reporting Act (FCRA) enforced by the Federal Trade Commission promotes accuracy in consumer reports and is meant to ensure the privacy of information in them. The GLB (The Gramm-Leach Bliley Act 2000) contains provisions for financial institutions to protect consumers' personal financial information and also requires all financial institutions to design, implement and maintain safeguards to protect customer information. In healthcare, the HIPAA (Health Insurance Portability And Accountability Act of 1996) Privacy Act limits who can look at and receive a patient health information and cites examples of the types of health information that is protected such as conversations on a patient's health by medical staff, information on a medical record and billing information amongst others. A new privacy law that loosely models the principles of consumer freedom and privacy rights in HIPAA and GLB is needed for the larger commercial realm. The only arena where privacy laws have extended to the commercial realm is with respect to data collection on children – the COPPA (Children's Online Privacy Protection Act**)** regulates the collection of personal information on children under 13 on commercial websites and online services.

The United States can use privacy laws in Europe and Canada as a model. The European Union (EU)'s Privacy Directive (effective since 1998) has tough rules on the protection of privacy of personal data. As a matter of fact, one of it's provisions discusses the special protection of sensitive data – data collection that identifies "racial or ethnic origin, political opinions, religious or philosophical beliefs or concerning health or sexual orientation is generally forbidden outright. Another provision of the EU directive requires companies to tell customers when they plan to sell their personal information to other firms. Companies in the US who inform consumers about the usage of personal data that is collected, do so on a strictly voluntary basis.

Canada too has a fairly recent privacy law AS OF January 2004; the PIPEDA (Personal Information Protection and Electronic Documents) for the private sector. Like the EU Privacy Directive, it too declares that all personal information that is collected must be used and disclosed, and stored securely. PIPEDA insists that people have the provision to alter any information collected on them and cites examples of information that is considered personal such as information include names, gender, contact information, blood types, income, credit records, loan records, disputes between a consumer and merchant, intentions to acquire goods and services and several others. Another striking difference of privacy legislation between Canada and the United States is the fact that

Canada has even appointed a privacy commissioner to oversee all privacy legislation unlike in the US which has no such equivalent role in government.

EU Privacy Directive at a glance: [11]

Personal information:	• must be processed fairly and lawfully. • must be collected and possessed for specified, legitimate purposes and kept no longer than necessary to fulfill the stated purpose. • must be accurate and up-to-date. • cannot be transferred to third parties without the permission of the individual providing the data, or data subject. In the case of data transfers across national boundaries, the Directive prohibits data transfers outright to any country lacking an "adequate level of protection," as determined by the EU. • that is identifying "racial or ethnic origin, political opinions, religious or philosophical beliefs . . . [or] concerning health or sex life." Under the Directive, such data collection or processing is generally forbidden outright. . • Is subject to be accessed and corrected for inaccuracies.

PIPEDA at a glance: [12]

Personal information must be:	• collected with consent and for a reasonable purpose • used and disclosed for the limited purpose for which it was collected • accurate • accessible for inspection and correction • stored securely

The Federal Trade Commission and other government bodies concerned with privacy need to examine the shortcomings of data privacy protection laws within the commercial realm. A privacy law that includes many similar provisions offered by the EU and Canadian privacy laws would greatly help enforce the national push towards privacy sensitive databases.

Conclusion:

The accelerating spread of production of ever cheaper and larger data storage space, the insatiable thirst to store as much data as possible, and the irresponsible disposal of unsanitized data media call for a more critical attitude toward storage of data. While this is a general problem, storage of personally identifiable data with negative social consequences is of especial importance. Wayner's proposal of *translucent databases* is an attempt in this direction and defining *privacy-sensitive databases* is a step to focus such attempts on a socially useful direction.

Securing private information is an important task that requires careful work. Such work needs to focus on the problem from a practical point of view, consider all implementation pitfalls carefully, and provide acceptable models that can be promoted. False assumptions on issues such as usability and security can give unwanted results.

While many of Wayner's techniques for preserving anonymity in databases are novel, they are rarely being used commercially today, as there is no primary driving force pushing for the privacy of personal data. In order for more rigorous data protection to take place, there needs to be a more aggressive campaign for privacy rights within the United States. Moreover, the level of awareness of system designers towards protecting personally identifiable data needs to be heightened, and database owners need to be made aware of their responsibility.

It is only through the awareness and cooperation of all involved that an acceptable level of protection of privacy can be provided.

References

[8] Ulf T. Mattson. A Database Encryption Solution That Is Protecting Against External And Internal Threats, And Meeting Regulatory Requirements, 8 July 2004

[9] Don MacVittie. Time is Right for Database Encryption, 9 Dec 2003

[10] Katherine Albrecht, Ed. M. Supermarket Cards: The Tip of the Retail Surveillance Iceberg

[11] Official version of Privacy Act. The European privacy Directive. 1995.

[12] Office of the Privacy Commisioner of Canada. The official version of PIPEDA.