CRY-HELP: Literature Synthesis

Submitted by Tami Maiwashe

ABSTRACT

For a long time, the focal point of research in database security has been access control – ensuring that only authorised people view an organisation's data. With increasing occurrences of insider attacks and a movement towards databases as a service though, database encryption has become an important area of research in database security. This technique aims to render a database unreadable to any unauthorised individual, such that even if an attacker were to bypass the access control system in place, they would be unable to view the data. But because of how expensive cryptographic computations generally are, for the past decade or so, database encryption has mostly just been a theoretical discussion constructing a design framework for schemes that would achieve it. However, recently CryptDB was put forward as a practical database encryption scheme. This paper benchmarked CryptDB against some of the design framework that was built in earlier discussions, and found that on the most part, it has not faltered; it has indeed fulfilled several researchers' design requirements for a database encryption scheme.

INTRODUCTION

Despite South Africa's crime statistics depicting a decrease over the years (Crime Stats SA, 2012), it must be made clear that this is an indication only of the levels of reported crime. While research in other countries shows that it is typically 'less serious' crime that people choose not to report to the police (Bennett & Wiegand, 1994; Skogan, 1977), in South Africa, this is not the case. In 2011, SAIRR published a report which announced that over half the crime committed in that year had not been reported to authorities, and these crimes included actions as harmful as house burglary and car hijackings (South African Institute of Race Relations, 2013).

A number of reasons have been suggested to explain this behaviour, but when Lasley (1995) investigated how the method of crime reporting affects one's decision to notify the police of a crime, he found that providing a computer-based approach (as opposed to the traditional telephonic approach) elicited more reports from people. He then put forward that perhaps providing (a greater sense of) anonymity in reporting a crime could reduce the number of unreported crimes.

That being said, developing a mobile application to support anonymous crime reporting will very likely be good and well, opening a safe channel for crime as rife yet kept secret as domestic violence and rape to be reported (1 Billion Rising, 2013), but unfortunately pointless if the sensitive information in the reports cannot be kept securely in a database.

In recent times, database security research has surged past its previous focus on access control, with effective solutions like role-based access control well established and in use world-wide (NIST, 2012). Growing interest is being placed in using encryption to secure the database itself, so that even if an unauthorised person is to evade the system's access control mechanisms, they won't be able to make sense of the data stored in it (Bouganim & Guo, 2009).

This paper reviews various design requirements that researchers have suggested for database encryption schemes, focusing on comparing CryptDB – a database encryption scheme recently put forward by Popa et al. (2011) – to them.

CRYPTDB AND HOW IT FARES AGAINST THE DESIGN FRAMEWORK

In their discourse about the various questions to be answered in building a database encryption scheme, Bouganim and Guo (2009) describe three different levels on which the encryption could be done: storage level, database level, and application level. They claim that encrypting in the storage subsystem has the advantage of not having to alter the front-end application, but the disadvantage is that this system cannot

tell between the different database objects and its structure, and as such, the encryption scheme would be unable to incorporate user privileges (for example, encrypting data that a user has access to under a key unique to them) and to vary encryption according to how sensitive certain data is.

They argue that encrypting on the database's level, on the other hand – that is, encrypting data as it is inserted in or retrieved from the database – would overcome these limitations as the database's design and structure are recognised there. The downside to encrypting on this level would be having to alter the front-end application to accommodate this 'new' database management system, and the application would not perform as quickly as before because of the overhead added by the encrypting/decrypting done at each insertion or retrieval of data.

Lastly, they suggest leaving the encryption and decryption to be done by the front-end application, stating the benefit of having the encryption keys kept separately from the encrypted data (an attacker would thus not find all the information needed to access the data in one place, as would be the case in the previous two proposals). But again, they highlight a number of disadvantages to this approach, including how the application would have to be changed and its performance negatively affected.

In light of all of this, Popa et al. (2011) chose to modify the traditional database management system structure slightly and add a component called a database proxy between the application server and the database server. This component handles the encryption and decryption of data. By adding this component, they successfully navigated the minefield described by Bouganim and Guo (2009), and found a level on which encryption and decryption could be carried out without having to change the application or the database software. This ease of integration with current database management systems is a feature that Shmueli et al. (2009) suggested a good database encryption scheme should have too.

Another factor that several papers (Bouganim & Guo, 2009; Shmueli, Vaisenberg, Elovici, & Glezer, 2009; RSA Security, 2013) agreed should be carefully considered in the design of a database encryption scheme is that of keys management. In fact, Bouganim and Guo (2009) went as far as to claim that such a scheme could only be as strong as the protection of its encryption keys. They stated that the important aspects of managing the keys are their location and the restriction of access to them. Shmueli et al. (2009) added key recovery as a crucial element of keys management, reasoning that a lost or damaged key renders the encryption of data useless.

In their paper, Popa et al. (2011) do not address the issue of key recovery, but explain that the encryption keys of the system are linked with users' passwords. They show that such a keys management system ensures that even if an attacker were to access the database, they would be unable to view the data to which logged-out users have access. However, the same can evidently not be guaranteed for users logged into the system at the time of the attack, as the data they're authorised to see might be decrypted already.

Bouganim and Guo (2009) further considered the assurance of data integrity, authenticity, and correctness as crucial parts of a database encryption scheme. However, the authors of CryptDB decided against extending their scope to cover those security issues (Popa, Redfield, Zeldovich, & Balakrishnan, 2011), so, for instance, a database administrator could delete data without a trace – this also goes against what Shmueli et al. (2009) had desired for a database encryption scheme.

Shmueli et al. (2009) recognised that encrypting a database would be an expensive operation, but outlined four areas where the overhead could be reduced, namely: the scheme should only encrypt sensitive data, and only data required for the execution of a query should be decrypted. CryptDB meets this requirement as it can encrypt just certain columns of the database. Over and above this, it mostly uses symmetric-key encryption, which is the more efficient choice for an encryption technique; as a result, the encryption scheme's performance overhead is even lower (Popa, Redfield, Zeldovich, & Balakrishnan, 2011). With

respect to the execution of queries, Popa et al. (2011) explain that CryptDB uses SQL-aware encryption to allow for the system to execute queries on encrypted data. In their paper, they illustrate how they developed SQL-aware encryption, which is simply an adaptation of well-known encryption schemes to the primitive operations that are at the core of database languages like MySQL (operations such as checking for the equality or the ordering of elements).

CONCLUSION

For roughly a decade, great discussion has surrounded the idea of database encryption in a bid to find a way to secure data beyond the reach of access control. This discussion has mainly been theoretical, with researchers hypothesising about what features a good database encryption scheme should have. CryptDB is a recently-developed practical database encryption scheme which this paper aimed to analyse with respect to some standards that arose from the decade's discussions. Essentially, the framework of standards focused on various aspects of the scheme's performance and the management of its keys. Based on these standards, researchers can be assured that CryptDB is a good database encryption scheme.

BIBLIOGRAPHY

1 Billion Rising. (2013). *TEARS (Transform Education About Rape and Sexual Abuse)*. Retrieved May 3, 2013, from One Billion Rising: http://onebillionrising.org/page/event/detail/w8g

Bennett, R. R., & Wiegand, R. B. (1994). Observations on crime reporting in a developing nation. *Criminology* , pp. 135-148.

Bouganim, L., & Guo, Y. (2009). Database Encryption. In S. Jajodia, & H. van Tilborg, *Encyclopaedia of Cryptography and Security* (pp. 1-9). Springer.

Crime Stats SA. (2012). *Crime stats simplified*. Retrieved May 3, 2013, from Crime Stats SA: http://www.crimestatssa.com/national.php

Lasley, J. R. (1995). When computing goes high-tech: an experimental test of computerized citizen repsonse to crime. *Journal of Criminal Justice* , 519-529.

NIST. (2012, June 11). *Role Based Access Control and Role Based Security*. Retrieved May 3, 2013, from NIST Computer Security Resource Center: http://csrc.nist.gov/groups/SNS/rbac/

Popa, R. A., Redfield, C. M., Zeldovich, N., & Balakrishnan, H. (2011). CryptDB: Protecting Confidentiality with Encrypted Query Processing. *SOSP* (pp. 1-16). Cascais: ACM.

RSA Security. (2013). *Securing Data at Rest: Developing a Database Encryption Strategy.* Retrieved May 3, 2013, from RSA Security: http://www.rsa.com/products/bsafe/whitepapers/DDES_WP_0702.pdf

Shmueli, E., Vaisenberg, R., Elovici, Y., & Glezer, C. (2009). Database Encryption - An Overview of Contemporary Challenges and Design Considerations. *SIGMOD Record* (pp. 29-34). Providence: ACM.

Skogan, W. G. (1977). Dimensions of the dark figure of unreported crime. *Crime & Delinquency* , 41-50.

South African Institute of Race Relations. (2013, March 19). *Over half of crimes go unreported.* Retrieved May 3, 2013, from South African Institute of Race Relations: http://www.sairr.org.za/media/media-releases/Over%20half%20of%20crimes%20go%20unreported..pdf/view