Cryptoid: Mobile Encryption

Michael Adams

Geysa Fernandes

Gregory Macri

May 9, 2016

Abstract

As we continue to develop hardware in more compact space, the field of mobile computing has and continues to expand on an exponential level. Though, as we develop smaller and faster hardware and a diverse body of software for these mobile devices, we are continually plagued with security issues. Computer securitys importance continues to grow as more and more people gain access to some form of a computer and more people are exposed to the possibility of sensitive data being stolen from their devices. We will focus on the use of encryption and biometrics as means to protect our data. We will answer what encryption is and how different forms of encryption work today. We will purpose a form of encryption that is deeply intertwined with that of the operating system. Instead of thinking as the operating system as a separate entity from our encryption, we purpose a form of encryption that continually exists as a standard function of our operating system.

Introduction

Encryption can be defined as the translation of data into secret code with the intent that it can later be deciphered by people authorized to do so. Silberschatz, Galvin and Gagne describe how an encryption algorithm involves five basic components: a set of keys, a set of messages, a set of ciphertexts, an encryption function, and a decryption function (Silberschatz, 652). The user encrypts their message using the generated key or keys into ciphertexts. Ciphertext is then sent through some channel, and once received, a decryption function also uses a key or keys to convert the ciphertext back into the original message.

There exist two forms of encryption algorithms, symmetric and asymmetric encryption. As the book describes, In a symmetric encryption algorithm, the same key is used to encrypt and to decrypt (Silberschatz, 652). The obvious security flaw here is that if someone trying to decrypt this information with malicious intent, they would only need to find the one key, and if found, would have access to all the encrypted information using that key. For this reason, we turn to asymmetric encryption algorithms.

With an asymmetric encryption algorithm there exist separate keys for encryption and decryption (Silberschatz, 654). The most common form of asymmetric encryption is public-key encryption. With public key encryption, the receiver creates two keys, a private key to decrypt that only it knows, and a public key made available to any sender that allows for the sender to encrypt what it wants to send (Silberschatz, 654). In this situation, we remove the security risk of having to send a key along with the encrypted data; instead, the key always stays securely with the receiver of the encrypted data.

A biometric is a unique, measurable, biological characteristic used for automatic recognition or identification of a human being. A measurable biological characteristic can be a fingerprint, iris, retina, and more recently, hand, face, voice, keystroke (typing rhythm), DNA and body odor.

Biometric characteristics are frequently used for security purposes. A biometric system can have two different objectives: identification and authentication. In an identification biometric system, the process is a one-to-many search of one biometric pattern through the entire stored data. In this case, we want to find the identity that corresponds to a specific biometric pattern. In an authentication biometric system, the process is a comparison between the entered pattern and one specific pattern in stored data corresponding to the claimed identity. In this case, the objective is to verify the authenticity of the identity claim. This process is also termed one-to-one search.

In general, the security of an encryption method is dependent on the security of the secret or private key. It is clearly not possible for a user to remember a cryptographically-strong key, so instead the user is asked to choose a passcode to decrypt the cryptographic key stored in the system. The problem is that the security of the cryptographic key is only as good as the security of the passcode. And sadly passcode authentication systems cannot distinguish between a legitimate user and a fraudulent user.

Biometric authentication offers an alternative to passcode authentication. Rather than entering a passcode to release the cryptographic key, the user will be prompted to provide a biometric sample as proof of the identity claim. The biometric authentication provides convenience to the user, as he doesnt have to remember and secure a passcode, and provides a unique and secure identity confirmation.

This process is called biometric encryption. It is important to note that biometric encryption is the process of managing the cryptographic key access. It does not directly provide encryption/decryption methods, but offers an alternative to passcode protection protocols. Biometric encryption is used to complement the security of existing cipher systems.

Methods of Encryption

A diverse body of encryption algorithms is used today across different platforms and industries. In this section, we will examine some of the more popular encryption algorithms. We will also discuss encryption techniques such as full disk and file-system level encryption.

One popular encryption algorithm that exists today is the Advanced Encryption Standard (AES), aka Rijndael. AES was developed by Belgian scientists Vincent Rijmen and Joan Daemen (Schwartz). It is a symmetrical encryption algorithm that generates a key size of 128, 192, or 256 bits. This algorithm was named as the Advanced Encryption Standard in 2001 by the National Institute of Standards and Technology (Schwartz).

Another popular form of encryption often used for communication is RSA encryption, which was developed by Ron Rivest, Adi Shamir, and Leonard

Adleman in 1978, and the acronym RSA stands for their last names. RSA is a form of public key encryption, which is a form of asymmetric encryption. This essentially means that there is a public key being given out to people who are supposed to read the encrypted text, and a private key which is not given out that encrypts the text. Although it is a very old technology, RSA and modifications of it are still in use today due to its strong mathematical basis.

Another popular form of encryption is Blowfish. Blowfish, unlike RSA, is a form of symmetric key encryption, developed by Bruce Schneier in 1993. Despite losing to Rijndael as the Advanced Encryption Standard, Blowfish is still a very useful algorithm. It makes use of Feistel networks, similar to DES/Lucifer, whereas AES does not. It is also a public domain encryption algorithm according to its creator, Bruce Schneier. It is currently used by many companies around the world to protect their data.

An older and increasingly less popular algorithm we considered is Triple DES. Triple DES solved many of Lucifer's weaknesses, but for many people, the damage had been done. DES/Lucifer was developed by IBM, and for a long time, it was considered to be the best algorithm, but as technology advanced and weaknesses were found, it quickly became useless. It was replaced by many algorithms, one of which is 3DES, or Triple DES, which is quite literally DES applied 3 times. 3DES, first saw widespread use around 1998, and by that time there were many superior algorithms already in use, but it still sees use occasionally.

Last but not least, we look at Twofish, the successor of Blowfish, also developed by Bruce Schneier. Like Blowfish, Twofish is a very solid and useful algorithm and in the public domain it is a fast and secure algorithm, it has always been a contender for the spot of best encryption algorithm. Twofish was built to be better than everything else available at the time, and while it is not the AES (Rijndael), it is still a very solid algorithm.

Many forms of encryption focus on encrypting data at its source before transmission over a network interface and decrypting it at the destination. Others, like disk encryption, focus on doing that internally on a machine, so the data located on that machine can only be accessed by whoever is authorized to do so. It is an increasingly popular form of encryption, and has been in the news, including the recent FBI vs. Apple dispute (Kharpal). It is essentially a means to guarantee privacy and security through the reversible scrambling of information, such that if any unauthorized user attempts to gain access to that information, they will be unable to do so. There remain weaknesses in such a system, because often times they are password protected, and as hence depend on the strength of the selected password, and passwords are inherently weak because they depend on people remembering them.

One form of disk encryption is filesystem encryption, where the filesystem layer of the operating system is doing the encrypting, and the disk itself is not being encrypted. This is not as common as full disk encryption, but it is useful all the same. It also shares a majority of the same weaknesses, although it may not quite be susceptible to catastrophic failure in the same way that full disk encryption is, where if the password fails, the entire system fails. For filesystem encryption, you can have multiple passwords for multiple files, albeit that may not be as efficient for the basic user, it is however slightly more secured.

OS Entangled Encryption

Now that we have examined the many types of computer encryption through different hardware and software encryption, let us examine a different approach that we propose as a way for computers, specifically mobile devices, to encrypt and decrypt data.

Our understanding of current models of encryption and decryption is that data is encrypted, and once received, given the proper authorization; the encrypted information is taken out of an encrypted state for the user to read, write, and otherwise manipulate the data available in some form. But what if the information once encrypted stayed in a constant form of encryption that the operating system decrypted in real-time? The idea is that the operating system would be able to keep data encrypted and be able to handle encrypted data and other system critical information while only decrypting it for context specific purposes. This technology coupled with that of biometrics, would allow for exponentially higher security when it comes to user data and the security of an operating system.

There are obvious pros and cons to this approach. For instance, researchers at Princeton University found that by having physical access to a device and by using a technique called Cold Booting, they could extract encryption keys from memory and use those keys to decrypt information that was encrypted on disks (Halderman). Through our approach, Cold Booting would not work because the information at and below the OS level would continually be encrypted, thus physical access to the device and the use of Cold Booting would not be effective. Another pro of the system is that because data and critical system information is all continually encrypted, any outside attacks from malicious code would be more easily detectable because it wont be in an encrypted form that the OS would recognize as safe. Because anything trying to interact with the OS, either data or processes, has to go through a level of checks before encryption, any malicious processes would be detected before any harmful interactions with the operating system could occur. We must also examine the drawbacks of such an implementation, the biggest of all; performance.

Research Methods and Results

With an operating system having to continually interpret encrypted data, we will see increased latency in the operating system. Though we have continually seen an increase with performance in technology, the feasibility of such a system still comes into question. For this reason we have come up with a research method that allows us to quantify exactly how expensive of a cost encrypting and decrypting a string of varying length is on a mobile device. We have written an android application that allows us to encrypt a string, and then decrypts it, and times how long it takes, before displaying every step. This can be run on any android device with at least Android Lollipop 5.0.1, enabling us to collect data from varying devices, such as tablets, smartphones, and android TV boxes.

The data we have collected unfortunately confirms our hypothesis that the burden on the system greatly increases, and applying it to every memory transaction would tremendously slow the system down. According to our data, this cost varies from about a 1/100th of a second to about a 1/8th of a second depending on the device capabilities. On top of the regular time it takes for the system to access memory, to read data, or write data, there would be an additional amount of time between 1/200th of a second and 1/2 second added onto it. The time is split due to the fact that encryption actions and decryption function similarly in AES when operating on single blocks whereas when operating on multiple blocks, decryption often takes a lot less time. This is based on the very worst case scenario of single block operations. This may not seem like a lot for a single memory transaction, but in the context of Android device usage, many programs may be in use which would take up a lot of memory. For instance, say if your favorite Android app stores about 250mb in memory. The overhead for that many transactions would make the application brutally inefficient. We must also include the device's operating system as well. If the operating system needs a large amount of memory to run, then this would mean that it would take a long time to put everything it needs to put in memory into memory, as well as read from it.

With that very expensive cost, is it an impossible ambition to encrypt/decrypt everything that comes out of memory? We have considered many solutions for this issue, ranging from hardware solutions to software solutions and filtration. Of the hardware solutions, one such solution is that another processor could be added to the device and put in charge of handling all memory transactions in parallel with the normal processors. This would mask the time these memory transactions take, similar to how a lever for opening a gate in a video game masks the loading time of the area located just beyond that gate. So, if we can successfully mask the time it takes for it to happen, we cannot guarantee that it still won't take a long time, if for no reason other than simply being unable to test this conclusion.

Another potential solution to the problem of minimizing the time it would take to do memory transactions with built-in encryption is to consider whether or not we need to encrypt every single piece of data. Do we really need to decrypt/encrypt this loop index every time the user compares it to something or increments/decrements it? If not, we need to be able to make that distinction. The distinction between necessary data and unnecessary data could prevent a lot of time wasting encryption. So how would we solve this in a way that would facilitate the safety of our system? The answer to that question lies in one of two possible solutions. We could impose a "virtual" memory layer, where memory we choose not to encrypt is located and kept track of through means of virtual addressing and some sort of algorithm which keeps track of how many times different chunks of memory changes themselves. This is not a very good solution to this problem because there is no real way for the system to determine if this data is actually important or not, even though this method would be very good for finding and excluding things like loop indices from encrypted memory. A responsible solution to the above problem would likely involve cooperation from the programmer, by implementing a way for the programmer to safely or unsafely store their variables. The programmer could be relied on to unsafely store their loop indices, because not doing so could negatively affect their programs. Other than the programmer, language developers could include some sort of compiler optimization to handle this as well.

Conclusion

We have begun to see an integration of strong encryption technologies (disk encryption or biometrics) in Apple and Android devices. Android provides full disk encryption software for its current devices and Apple has integrated biometrics into its newer iPhones through thumb print authorization.

After our investigation and very small scale testing of a single aspect of the system; what is the possibility that such a system like the one we proposed will exist in the future? If we recognize a continuing trend of processing power with the growing field of parallel computation, I think we will begin to see likeminded implementations of operating systems as processing power becomes less and less of an issue and as security begins to dominate the field of computer hardware and software. As a society, we will begin to exchange the costs of processing power for security as we become more connected and continue to invest our trust in things like mobile devices in being able to protect very sensitive data. Though for this given research paper, we are limited in time and thus cannot expand upon this idea beyond the high level description we have given along with the obvious issues we would run into with development of such a system. Though, given the opportunity we may pursue some aspect of this project at a later time.

Appendix

Viewing the project in a more global sense, we wanted to focus and reflect on the learning aspects of this project. From this project we have been able to develop team building skills and have learned to more easily work with others. We have seen the benefits of working with others as we have continued to motivate each other in finishing our work as deadlines close in as well as clarify any issues we have come across as we trudged through the development of our application and paper. Through this project we have studied different aspects of encryption and biometrics that have greatly expanded our knowledge in the field of computational security. We have also learned skills in developing for android devices as well as a broad idea of how encryption algorithms are integrated into software. This project has allowed us to expand and explore how operating systems work as the information was needed to better develop our initial idea. Overall this project has been a great learning experience and as such has better equipped us for projects like it in the future.

Bibliography

Silberschatz, A., Galvin, P. B., & Gagne, G. (2014). *Operating System Essentials* (2nd ed.). Danvers, MA: Wiley.

Schwartz, J. (2000, October 03). *U.S. Selects a New Encryption Technique*. Retrieved May 01, 2016, from http://www.nytimes.com/2000/10/03/business/technology-us-selects-a-new-encryption-technique.html

Halderman, J. A., Schoen, S. D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J. A., . . . Felten, E. W. (2009). Lest we remember. Communications of the ACM Commun. ACM, 52(5), 91

Blowfish encryption. (2014). Retrieved May 01, 2016, from http://www.splashdata.com/splashid/blowfish.htm

Rivest, R. L., Shamir, A., & Adleman, L. (1983). *A method for obtaining digital signatures and public-key cryptosystems*. 1-15. Retrieved April 30, 2016.

Karthik, S., & Muruganandam, A. (2014). *Data Encryption and Decryption by Using Triple DES and Performance Analysis of Crypto System*. International Journal of Scientific Engineering and Research, 2(11), 24-31. Retrieved April 30, 2016.

Bragg, R. (n.d.). The Encrypting File System. Retrieved May 01, 2016, from https://technet.microsoft.com/en-us/library/cc700811.aspx

Cavoukian, A., & Stoianov, A. (2007). Biometric Encryption: Technology for Strong Authentication, Security and Privacy. 1-48. Retrieved April 18, 2016, from https://www.ipc.on.ca/images/resources/bio-encryp.pdf

Soutar, C., Roberge, D., Stoianov, A., Gilroy, R., & Kumar, B. V. (1998). Biometric Encryption. 1-28. Retrieved April 18, 2016, from http://www.cse.lehigh.edu/prr/Biometrics/Archive/Papers/BiometricEncryption.pdf

Kharpal, A. (2016, March 29). Apple vs FBI: All you need to know. Retrieved May 09, 2016, from http://www.cnbc.com/2016/03/29/apple-vs-fbi-all-you-need-to-know.html