Volume:04 Issue:01 Pages:1503-1508 (2012) ISSN : 0975-0290

Accessing a Network using a Secure Android Application

Padmalatha Ragunathan
Dept of Information Technology, Amrita School of Engineering, Coimbatore, India
Email: padmalatha.ragu@gmail.com

Kishore Sambath
Dept of Information Technology, Amrita School of Engineering, Coimbatore, India

Vishnu Karthik L
Dept of Information Technology, Amrita School of Engineering, Coimbatore, India

---**ABSTRACT**---

Security plays a vital role in today's mobile world. There are security issues like sniffing of data while accessing information through open channel. Proper security measures can help to deal with the common security threats faced by mobile phone users such as data protection, privacy, application and personal information security. Cryptographic techniques play an important role in protecting communication links and data, since access to data can be limited to those who hold the proper key. This paper discusses a method to securely access information in a network by an android mobile application using AES cryptographic technique. The paper describes a new key sharing algorithm, based on the symmetric key management, for faster and efficient encryption of data that is suitable for use in a mobile device.

Keywords – **Data security, Data vulnerability, Methods of securing Android application data.**

Date of Submission: April 10, 2012 Date of Acceptance: May 31, 2012

I. INTRODUCTION

The rapid growth of portable electronic devices with limited power and memory has opened a vast area of mobile computing and challenges for implementing security in such devices which are always connected to the internet. Smart phones that are powered by either Android OS, iOS, Web OS, Bada OS, BlackBerry OS or Windows Phone OS are examples of portable electronic devices that are becoming an integral part of everyday life.

Android is a privilege-separated operating system, in which each application runs with a distinct system identity that helps in identifying and isolating application resources [1]. This forms the application sandbox. However, vulnerability exists with many Android devices that would allow malicious apps to bypass the permissions request process and tap into user's personal information, without the knowledge of the user.

Many encryption algorithms are widely available and used in information security. They can be categorized into symmetric (private) and asymmetric (public) key encryption. In Symmetric keys encryption or secret key encryption, only one key is used to encrypt and decrypt data. The key should be distributed before transmission between entities. Key plays an important role. If weak key is used in algorithm then everyone may decrypt the data. Strength of Symmetric key encryption depends on the size of key used. For the same algorithm, encryption using longer key is harder to break than the one done using smaller key. Some examples of such algorithms are RC2, DES, 3DES, AES, etc. [2]

Asymmetric key encryption or public key encryption is used to solve the problem of key distribution. In Asymmetric keys, two keys are used; private and public keys. Public key is used for encryption and private key is used for decryption (E.g. RSA and Digital Signatures). Because users tend to use two key: public key, which is known to public and private key which is known only to the user. There is no need for distributing them prior to transmission. However, public key encryption is based on mathematical functions, computationally intensive. [2]

The later part of the paper includes as follows: Section 2 formulates the points regarding security in Android mobile platform. Section 3 presents the importance of secure apps and the working of our application, which helps in blocking calls. Section 4 provides the study of cryptographic algorithms such as RSA, ECC and AES. This paper describes AES algorithm with a new key sharing method as an appropriate technique for securely accessing a network from an Android mobile device. Section 5 formulates the points regarding AES implementation details in mobile as well as web modules of the application with the new key management algorithm. Section 6 presents the results. Section 7 concludes the paper.

II. Android mobile platform

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. The Android SDK provides the tools and APIs necessary to develop applications on the Android platform using the Java programming language.

The main building blocks of Android platform are device hardware, operating system and application runtime. The Android application sandbox isolates data and code execution on a per-application basis [3]. Android application frameworks use robust implementations of common security functionality such as cryptography, permissions and secure IPC.

On Android, the Dalvik Virtual Machine is not a security boundary – the application sandbox is implemented at the OS level, so Dalvik can interoperate with native code in the same application without any security constraints [4]. Thus we concentrate on secure Android applications.

III. The secure call blocker app

There is limited storage on mobile devices. If any client using a modular application to retrieve data using dynamic class loading from sources that are not verified, such as unsecured network sources or external storage, there are possibilities that information might include malicious behavior [5]. There are a few call blocking applications in android market. Our application, "Brigadier", is a secure call blocker application that is used to send, store and retrieve blacklisted contacts on the server on the internet.

1. Working of the App

- Whenever a call is received, it is first checked to see if it can be trustable or not.
- All the contacts are trustable along with some favorite (special) numbers.
- When a call is received from an unknown number, the app flags the number as untrusted. It then sends a message to that number with our protocol (send: details <name> <email>). The caller on the other hand has to respond to that message with the protocol.
- The app running at background will constantly poll to check for messages from the untrusted numbers. Once it has the details, the user is prompted to add the number to the trusted list.
- With the details received, optional information can also be fetched from the social networking sites such as Facebook.

2. Data connection and security

Once the user installs and registers his details with the app, login details would be provided. With the account login, the user can login into the app.

The user can modify the settings on how the blacklisted number should be handled – whether to block calls or messages or both. The app is designed in such a way that the user's blacklist can be synchronized to the server securely.

Since the Android mobile devices use limited storage and the data transfer occurs between server and client device on the internet, a highly efficient security algorithm that uses minimal key size than that of conventional Public Key Cryptographic systems has to be used.

3. Data leak vulnerability

Android is more exposed to vulnerabilities and malware attacks because of several reasons: the openness of the platform, multiple OEMs implementing the OS and the apps in separate ways and lots of application available in several sources.

Michael Grace, Yajin Zhou, Zhi Wang and Xuxian Jiang [6] have found that by simply clicking on a link, Android users may give attackers access to personal information. If exploited, the vulnerability would allow a malicious web site to read and upload contents of any file stored on the phone's microSD (memory) card. Information on the memory card could include saved voicemails, photos or online banking data, etc.

That is, with the ClientLogin protocol, applications request an authToken from the Google service by sending an account name and password via a HTTPS connection. The authToken is valid for up to two weeks and is used for subsequent requests to the Google service API. If the authToken is sent over unencrypted HTTP, an attacker could use network sniffing software, like Wireshark, to grab it.

Thus our focus is to improve security in Android mobile devices by developing an application that is secure and void of any data leak vulnerability.

IV. Cryptographic algorithms

Private Key algorithms with high throughput are suitable for data communication, while public key algorithms with much lower throughput are suitable for private key exchange and authentication. Among all available algorithms, RSA, advanced encryption standard (AES), and Elliptic Curve Cryptography (ECC) which are approved by standard organizations are selected for the study.

1. Rivest Shamir Adleman cipher

The RSA algorithm is based on the presumed difficulty of factoring large integers, the factoring problem [7]. Here, a product of two prime numbers is published along with an auxiliary value, as the public key. The prime factors must be kept secret. Anyone can use the public key to encrypt a message, but only someone with the knowledge of the prime factors can feasibly decode the message.

1.1 RSA vulnerabilities

- The system (N,D,E) is likely to be insecure if (p-1), for the p that is one of the factors of N, is a product of small primes. [8]
- When RSA is implemented with several key pairs, the implementers often choose to use the same N for all key pairs, thus saving computation time. However, since the private and public exponents together always assist in factoring N, every single member of the system will be able to factor N with his key pair and use that result to invert any public exponent to the corresponding private exponent. So it is necessary to generate a new N value for each key pair. [8]

TABLE I

Comparison of Performance Time (milli secs) of RSA and ECC at clock speed of 400 MHz [10]

System	Key generation	Signature	Verification	Total Time
RSA-1024	2,740.87	66.56	3.86	2,811.29
RSA-2048	26,442.04	440.69	13.45	26,896.18
ECC-163	1.47	2.11	4.09	7.67
ECC-233	3.11	4.03	7.87	15.01

2. Elliptic Curve Cryptography

Elliptic Curve Cryptography is an approach to public key cryptography based on the algebraic structure of elliptic curves over finite fields [9]. Its security comes from the elliptic curve logarithm, which is the Discrete Logarithmic Problem [12] in a group defined by points on an elliptic curve over a finite field. This result in a dramatic decrease in key size needed to achieve the same level of security offered in conventional Public Key Cryptographic schemes [10].

Tadeusz Struk [11] has done experiments on mobile devices with small PC hardware (ARM 32) and concluded that the operation of ECC is faster, with fewer processor cycles, than RSA. A faster operation means less heat and power consumption, longer battery life and small portable devices that run longer and still being able to provide equivalent security level. It is also stated that 160-bit key ECC implementation yields 4 times faster security than 1024-bit key RSA implementation.

Elliptic Curve Cryptography is based on the Discrete Logarithm Problem [12]. Unfortunately, there are no known polynomial time algorithms for finding a large number of points on an arbitrary curve [13]. Thus implementing the message imbedding technique for ECC is difficult.

2.1 Advantages

- Decreased key size.
- Better performance in mobile devices than RSA [10].

2.2 Disadvantage

- Difficulty in message imbedding.

3. Advanced Encryption Standard

Advanced Encryption Standard [14], also known as Rijndael, is a symmetric block cipher that uses cryptographic keys of 128, 192 and 256 bits to encrypt and decrypt data in blocks of 128 bits. It operates on a 4x4 column major order matrix of bytes and most of the calculations are done in a special finite field.

3.1 Advantages

- Faster than asymmetric key ciphers [15].
- AES 128 bit key usage is faster than ECC 256 bit key usage [16].

3.2 Disadvantage

- Key exchange problem.

Table I compares the performance time of RSA and ECC in milliseconds. Table II discusses the NIST guidelines of equivalent key sizes of ciphers. Table III shows the number of clock cycles required per cipher.

TABLE II

NIST Guidelines - Equivalent Key Sizes of Cryptographic Algorithms

ECC Key Size (Bits)	RSA Key Size (Bits)	Key Size Ratio	AES Key Size (Bits)
163	1024	1 : 6	
256	3072	1 : 12	128
384	7680	1 : 20	192
512	15360	1 : 30	256

TABLE III

No. of Clock Cycles Required Per Algorithm

Algorithm	Encryption	Decryption
AES	951	2036
ECC-163	3,414,850	

If key exchange problem could be overcome in AES, then implementation of AES would yield higher security than that of ECC or RSA.

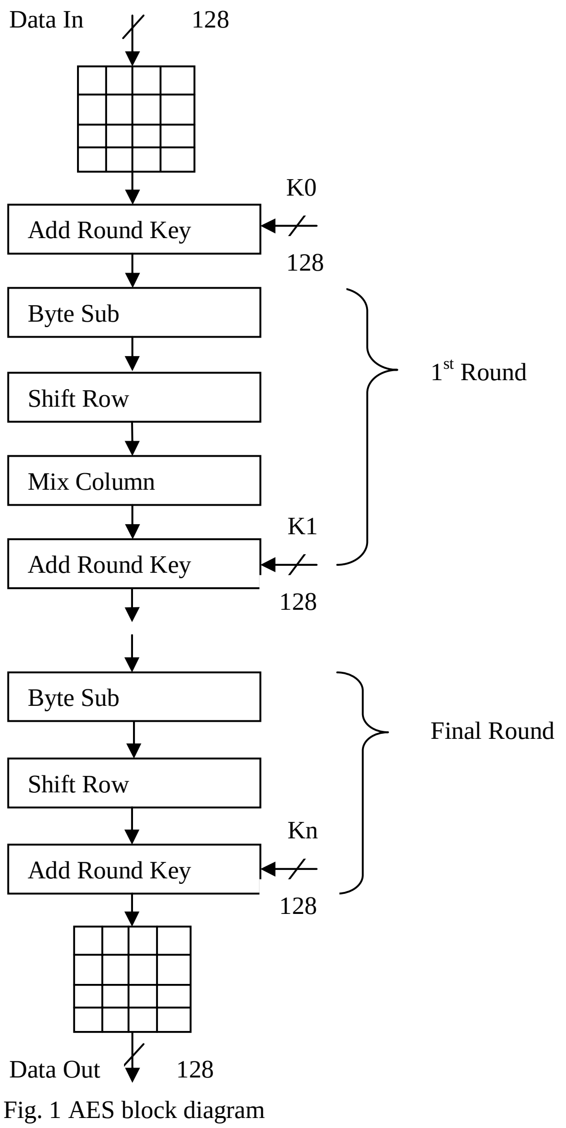

Fig. 1 AES block diagram

V. IMPLEMENTATION DETAILS

This section highlights the implementation techniques used in building the application. For our experiment, we use a notebook with Intel Core i5-2410M CPU @ 2.30GHz for running the server locally and HTC Wildfire having Android Froyo 2.2.1with ARM 11 CPU @ 528 MHz. Since we are concerned with mobile devices with limited storage and processing speeds, we have chosen AES with a new key sharing technique for computational efficiency.

To secure our Android app, "Brigadier", all data connections and data transfers from and to the client application, to and from the server has to be authenticated, encrypted when sent and decrypted when received using the AES algorithm that takes care of the key exchange problem. The app will help in blocking calls and messages from untrusted contacts stored as blacklists in the client application which are synchronized to the user's account in the server.

For the AES algorithm, the length of the input block, the output block and the state is 128 bits. This is represented by $N_b = 4$, which reflects the number of 32-bit words. The length of the cipher key, K, is 128 bits which is represented by $N_k = 4$. The number of rounds to be performed during the execution of the algorithm is dependent on the key size which is represented by $N_r = 10$.

The block diagram of the cipher is described in the Fig. 1. The AES algorithm takes the cipher key, K, and performs a key expansion routine to generate a key schedule. The cipher transformations can be inverted and then implemented in reverse order to produce a straight forward inverse cipher for AES algorithm. The individual transformations used in the inverse cipher – InvShiftRows ( ), InvSubBytes ( ), InvMixColumns ( ) and AddRoundKey ( ). Fig. 2 describes the data flow diagram for "Brigadier". The user data includes the blacklisted contacts which includes a name and a number.

1. Key sharing algorithm

The mobile app and the server will already have a key, K_{EM}, is 128 bits which is hardcoded. For the authentication of the user, MD5 hashing technique is used which is of length 128 bits. The hashed password of the user (application) is represented as K_P.

In order to generate round keys for the AES technique, an initial key, K_I, has to be obtained. This is key can be formed by XOR-ing the even bits of the hashed key, K_P, with the embedded key, K_{EM}. Equation (1) represents the same.

$$: \quad K_I = K_P \text{ (even bits)} \bigoplus K_{EM} \quad (1)$$

Thus K_I is the 128 bits symmetric key for that user. So every user would generate his secret key only at the runtime. The key is not stored anywhere. The XOR-ing provides randomness to the static embedded key, K_{EM}.

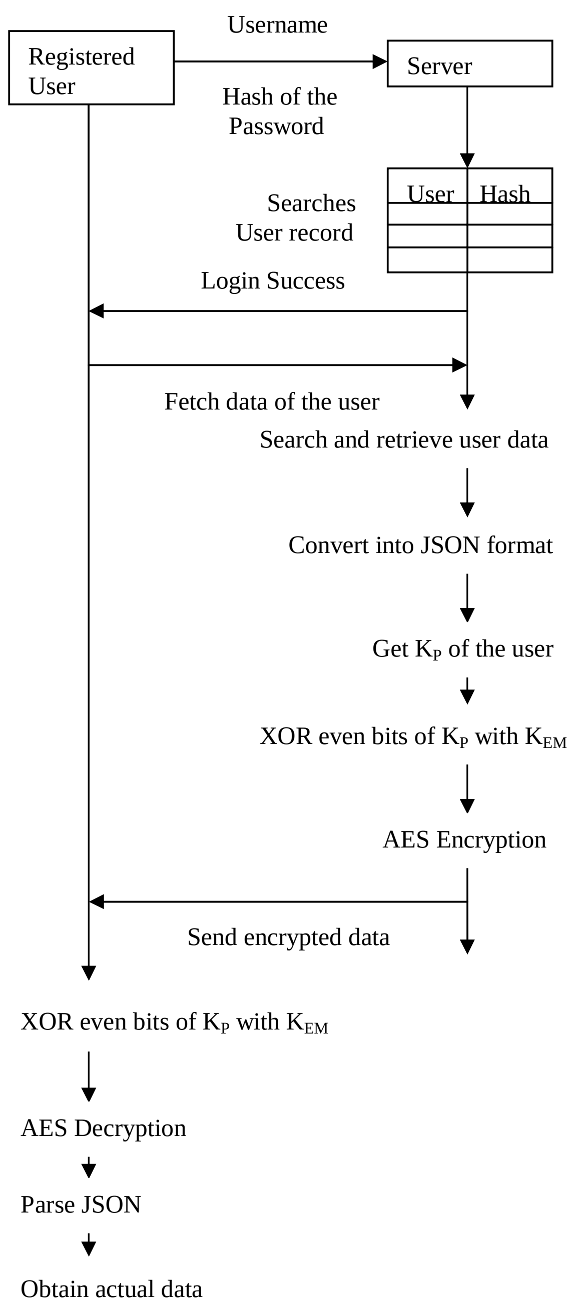

Fig. 2 Data flow diagram for Brigadier

VI. Results

The Android call blocker application, "Brigadier", is implemented using Android SDK in Eclipse IDE and requires Android version 2.2, Froyo, and up. It is found that security of the transmission data (blacklisted contacts) is ensured by implementing MD5 – 128 bits hashing for authentication, a faster yet efficient algorithm, AES, for cryptographic encryption and decryption using 128 bits key and the new key sharing algorithm. Every user has a unique key generated at runtime which is not stored anywhere. A secure web based client engine is also implemented and hosted in the internet. From the implemented experimental result, the time required in milli seconds in various stages (i.e. key generation, encryption, decryption and total time) of ECC, RSA and AES with our new key management algorithm is shown in Fig. 3.

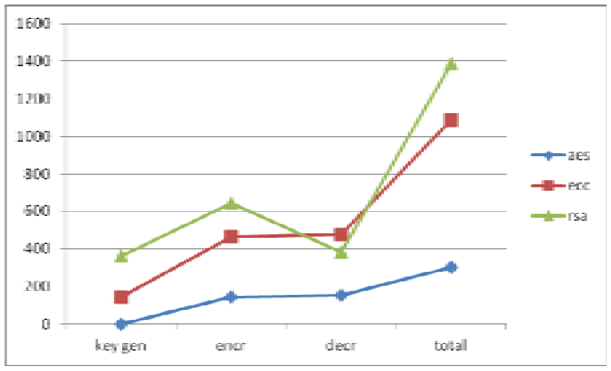

Fig. 3 Comparison of cryptographic algorithms (time in milli seconds)

VII. Conclusion

This paper presents a design of a cryptographic technique for accessing a network using a secure Android application. We achieved this by expressing the importance of a faster yet efficient algorithm, AES, for security in mobile devices in terms of simple logical operations that maximize efficiency in using less processing cycles, power and memory used. Further improvements can be done by implementing larger key lengths of AES technique with modifications to lower processing cycles that may ensure consumption of minimum battery power.

References

[1] Google Inc., *Android security overview* http://source.android.com/tech/security/index.html

[2] D.S. Abdul Elminaam, H.M. Abdul Kader, M. Mohamed Hadhoud, Performance Evaluation of Symmetric Encryption Algorithms, *International Journal of Computer Science and Network Security*, *8*(12), 2008.

[3] Goolge Inc, *Designing for security* http://developer.android.com/guide/practices/security.html

[4] Google Inc, *Security Architecture* http://developer.android.com/guide/topics/security/security.html

[5] W. Enck, M. Ongtang and P. McDaniel, *Understanding Android security*, Pennsylvania State University.

[6] M. Grace, Y. Zhou, Z. Wang and X. Jiang, *Data Leak Vulnerability haunts Android*, North Carolina State University. fchttp://web.ncsu.edu/abstract/technology/haunted-android/

[7] DI Management Services, Australia. *RSA Algorithm* http://www.di-mgt.com.au/rsa_alg.html/

[8] Y. Kumar, R. Munjal and H. Sharma, Comparison of Symmetric and Asymmetric Cryptography with existing vulenerabilities and counter measures, *International Journal of Computer Science and Management Studies*, 11(03), Oct 2011.

[9] N. Kobiltz, Elliptic Curve Cryptosystems, *Mathematics of computation*,48(177), January 1987, pages 203-209.

[10] W. Chou, *Elliptic curve cryptography and its applications to mobile devices*, University of Maryland, College Park. Advisor: Dr. L. Washington, Department of Mathematics.

[11] T. Struk, *Elliptic Curve Cryptography as a suitable solution for mobile devices*, National University of Ireland, Galway. Advisor: Dr. M. Schukat. Mail: tstruk@gmail.com

[12] D. Hankerson, Dept of Mathematics, Auburn University., S. Vanstone and A. Menezes, Dept of Combinatorics and Optimization, University of Waterloo., *Guide to Elliptic Curve Cryptography*, Springer Publication.

[13] Padma Bh, D. Chandravathi, Asst. Prof, Dept of MCA, GVP College, Vishakapatnam., and P. Prapoorna Roja, Prof, SSN College of Engineering, Dept of IT, Chennai, *Encoding and decoding of a message in the implementation of Elliptic Curve Cryptography using Koblitz's method*.

[14] *Federal Information*, Processing Standards Publication 197.

[15] K. Kenan, *Securing databases with cryptography* http://www.informit.com/articles/article.aspx?p=423771&seqNum=2

[16] Prof. X. Deng and Dr. Duncan S Wong, *Elliptic Curve Scalar Multiplier (ECSM) IP core*, City University of Hongkong. http://www.cs.cityu.edu.hk/~ecc