dLocAuth: a dynamic multifactor authentication scheme for mCommerce applications using independent location-based obfuscation

Torben Kuseler, Ihsan A. Lami
Applied Computing Department
University of Buckingham
Hunter Street, Buckingham, MK18 1EG, UK
(phone: 44-1280-814080; email: first.last@buckingham.ac.uk)

ABSTRACT

This paper proposes a new technique to obfuscate an authentication-challenge program (named LocProg) using randomly generated data together with a client's current location in real-time. LocProg can be used to enable any handset-application on mobile-devices (e.g. mCommerce on Smartphones) that requires authentication with a remote authenticator (e.g. bank). The motivation of this novel technique is to a) enhance the security against replay attacks, which is currently based on using real-time nonce(s), and b) add a new security factor, which is location verified by two independent sources, to challenge / response methods for authentication. To assure a secure-live transaction, thus reducing the possibility of replay and other remote attacks, the authors have devised a novel technique to obtain the client's location from two independent sources of GPS on the client's side and the cellular network on authenticator's side. The algorithm of LocProg is based on obfuscating "random elements plus a client's data" with a location-based key, generated on the bank side. LocProg is then sent to the client and is designed so it will automatically integrate into the target application on the client's handset. The client can then de-obfuscate LocProg if s/he is within a certain range around the location calculated by the bank and if the correct personal data is supplied. LocProg also has features to protect against trial/error attacks. Analysis of LocAuth's security (trust, threat and system models) and trials based on a prototype implementation (on Android platform) prove the viability and novelty of LocAuth.

Keywords: challenge response, mCommerce, mobile authentication, obfuscated interpretation, software protection

1. INTRODUCTION

The use of Smartphones to perform applications requiring client's authentication and secure environment (e.g. financial transactions) are increasing rapidly. In the US, for example, over 10 million clients used Smartphones for mobile banking in Q4/2010. This represents an increase of 120% from the year before [1]. However, because Smartphones are easily lost or stolen, reliable authentication and data protection are still major concerns in mobile applications, because PIN & chip used for authentication are open for hackers. As a result, financial companies restrict access, on average, to 70% of possible services to their clients via mobile apps [2]. Thus this paper focuses on securing the application as an added security prior to the actual agreed authentication between the two parties.

Authentication schemes are traditionally categorised into 3 groups [3]: knowledge-based (e.g. PIN), object-based (e.g. token) and identity-based (e.g. biometrics). Recently, further factors like “when (time) and where you are (location)” [4] have been proposed to strengthen the security of authentication systems. Smartphones are well suited for multi-factor authentication applications because they feature a variety of sensors / receivers to collect any required authentication data, e.g. camera to capture the user’s face, or a GPS receiver to determine current location and time.

Like other location-based multi-factor authentication schemes, the motivation behind LocAuth is, therefore, to restore some of the traditional face-to-face security characteristics missing in mobile transactions in a robust mechanism offered by obfuscation. Thus simulating the “good old” office-based authentication where the authenticator has the guarantee that the contract is signed by the known client at that particular place and moment in time. This eliminates attackers from pretending to be at a certain location as well as re-using previously gathered data of a genuine client.

The LocAuth scheme combines personal client data (e.g. PIN, to assure the genuine client), a stored Key-on-Phone (KoP, to ensure the correct device) with location and wireless communication properties (e.g. GPS / Serving cellular tower) to assure the client's position in real-time as well as randomly created nonce(s) to prevent replay attacks. LocAuth can be added to any existing mobile applications such as [5]. To make LocAuth more secure, the client's Smartphone location is independently determined on the authenticator side via the cellular network operator that serves the Smartphone using tower-based localisation techniques [6]. In contrast, at the client side, the location necessary to respond correctly to the bank's challenge is determined via the GPS receiver onboard the client's Smartphone. Therefore, if these two locations are within a certain range, and the client also provides the correct personal data, then the correct de-obfuscation key is generated.

The overhead and offered-security of LocAuth is analysed to assure its viability. LocAuth is tested by a prototype implementation on the Android platform (used in most Smartphones). Real GPS location and tower triangulation measurements in London are used in the experiments. This helped proving the concept and testing the practicality and accuracy of the location-key generation used. The obfuscated-program is designed to always run and terminate correctly, but shall calculate a wrong result when an incorrect key is used. This property is important for challenge/response techniques as it eliminates possible "try-and-error" attacks, because an attacker cannot decide, if the calculated result is correct.

The rest of the paper is organised as follows: Section 2 provides an overview of related work in software protection based on obfuscated interpretation as well as location-based security. Section 3 describes the concept of LocAuth, and section 4 illustrates the implementation, trials and security analysis. Section 5 concludes the achievements and outlines future work.

2. BACKGROUND AND RELATED WORK

2.1 Obfuscated interpretation and software protection

Software obfuscation [7] techniques protect a software program against unauthorised modifications. Software obfuscation makes it more difficult for an attacker to understand and change a program, for example through control flow obfuscation [8].

The first obfuscated interpretation [9] implemented a finite state machine (FSM) having a set of transition rules to convert the original program's instructions into new varied instructions which have no static relationship to the original ones. During program execution / interpretation, the obfuscated instructions are converted back into the original instructions based on a key. Therefore, an attacker is unable to understand the real semantics of the obfuscated program, even if the attacker has access to the code. This idea was then extended by introducing a software implemented permutation-based interpreter (PMI) that can be changed much easier [10].

The PMI framework was further extended by the oBiometrics software protection scheme [5] were the PMI key is replaced with a biometric generated key, thus removing the necessity of encrypting the permutation rules as well as securely-hiding the permutation interpreter and also tightly bound the program to the genuine user.

2.2 Location-based security and encryption techniques

Location proofs enable applications to check the current location of the user [11]. A proof can be a piece of data generated by a known and trusted sender (e.g. WiFi Access Point) that can be sent to the Smartphone on request. The Smartphone stores the received proof for immediate or later use and attaches it to a message in the case a location-proof is required. An example of such secure localisation and certification service that tags digital content, e.g. photos or communication messages with a location and time stamp was proposed in [12]. Other possible methods to determine the Smartphone position include GPS or WiFi-AP and were proposed for example in [13].

In contrast to LocAuth, these methods, as a first step, calculate the location on the Smartphone, and then, as a second step, the claimed location is verified by the receiver. A drawback of these techniques is that the location proof is stored on the Smartphone and thus, exposed to a malicious user. LocAuth overcomes this potential security problem because the LocAuth challenge is generated completely independent from the position determined by the user. The challenge location is based only on information received from the trusted network operator, as described in the trust model in section 3.

GeoEncryption [14] is a similar approach to LocAuth. This extends a classic hybrid cryptographic algorithm with a GeoLock functionality to ensure the location binding. Instead, GeoEncryption allows the message receiver to open (decrypt) the message only if the receiver's location is inside the required area. A practical mapping technique to convert a location to a unique key for the GeoEncryption was later described in [15].

3. CONCEPT AND SYSTEM/TRUST MODEL OF LOCATION-OBFUSCATED SOFTWARE PROTECTION

eCommerce and mobile banking are two typical applications targeted by this solution. LocAuth system model comprises 4 parties; a) the application provider (MAP) such as a bank who offers a service, b) the genuine client who uses the offered service and who has a previously established business relationship to MAP, c) the attacker who illegitimately wants to use the service, and d) the cellular mobile network operator (MO) who services the genuine client Smartphone and also communicates the genuine client Smartphone location to the MAP. This paper considers the following threat and trust model between the involved parties. The genuine client trusts MAP and MO. MAP trusts MO to determine and report the correct location of the genuine client Smartphone. I.e. this paper assumes a contractual relationship between MAP and MO. MO also trusts MAP in such a way, that MAP will not use the received client location for any fraudulent activities, i.e. breaches the client's location privacy. To protect the client's privacy, privacy preserving location verification techniques can be added to LocAuth [16]. MAP does not trust the location determined on the client side.

LocAuth authentication scheme starts on the client side by using the oBiometrics secured service [5] provided by the authenticator (Figure 1). The client starts the "protected application" on their Smartphone. A biometric key is then generated from a freshly captured biometrics and used to de-obfuscate the oBiometrics secured application.

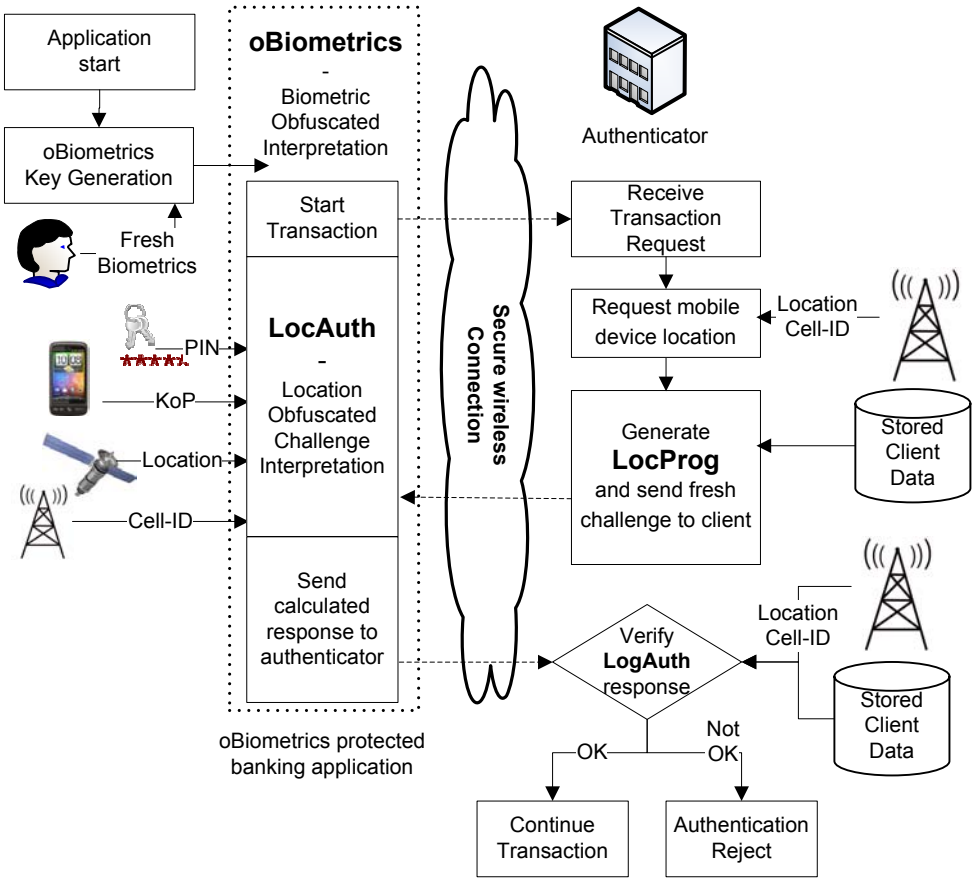

Figure 1. LocAuth algorithm for Location-Obfuscated Software Protection

If the fresh biometric feature vector of the client is correct, the biometric-obfuscation program interpretation is successful and an initial message is send to the authenticator. Upon receiving this message, the authenticator requests the current

location of the client's Smartphone, including the cell-ID used to serve this client's handset, from the MO. A fresh LocProg is then generated which is location-obfuscated by a key based on this client's location. LocProg contains random elements such as nonce(s), dynamic changing values known independently to both sides (e.g. cell-ID) and client data enrolled with the bank (e.g. PIN and a stored Key-on-Phone, KoP). These random elements act as multi-factors to prevent replay attacks and to assure the real-time (i.e. the response will only be accepted by the authenticator within a short time span, e.g. 5 seconds) and one-time property of this challenge. These dynamically changing values, together with the enrolled client data, also guarantee that only the genuine user with his own mobile device can calculate the correct response as discussed in the security analysis in section 4.3.

The obfuscation key is generated as described in section 4.1. The obfuscated LocProg is then sent back to the client. The application on the client's side receives the fresh LocProg and dynamically integrates it into the oBiometrics flow. After the client enters his PIN, the stored KoP is retrieved from the Smartphone memory or SIM card, and the current location of the Smartphone is determined by the GPS receiver onboard the handset, i.e. the client determines his location completely independent from the technique employed by the bank. The used cell-ID is then identified and the associated location-key is calculated. For correct calculation of the same location-key, LocAuth requires the client to be static or slow-moving during the process. The calculated key is then used to de-obfuscate the challenge interpretation and PIN, KoP and cell-ID become parameters to this LocProg. If the correct location key, PIN, KoP and cell-ID are used, i.e. the client is within a certain area around the location determined by the authenticator; the client is able to calculate the correct response. The correct response proofs to the challenger that the Smartphone is inside the correct area (location proof), that the response was calculated in real-time (nonce, dynamic changing value) by the genuine client (PIN) and Smartphone (KoP). If successful, the authenticator verifies the response and continues the transaction, otherwise authentication is rejected.

4. IMPLEMENTATION AND RESULTS

4.1 Key Generation for Location-Obfuscated Interpretation

The correct interpretation of the received location-based challenge requires, on the client side, the corresponding de-obfuscation keyDe-Obf to the obfuscation keyObf, used by the authenticator to create LocProg. To test the viability of LocAuth, location measurements (longitude and latitude values) were taken by the authors in London. Entries 1-6 in table 1 show some of these real measurements, for illustration. Entries 7-8 (shaded) are added intentionally as wrong control locations to reflect an attack on LocAuth that should result in being identified and rejected by the authenticator because the client (attacker) does not use / know the real location of the Smartphone.

Table 1. Location measurements in London.

	GPS-based location (L_C)		Tower-based location (L_A)			Distance between L_C and L_A, meter (L_E)	Cell-ID
	Lat M (deg)	Lon M (deg)	Lat T (deg)	Lon T (deg)	PV (m)		
1	51.4977	0.0080	51.497	0.007	300	104	53209247
2	51.5142	-0.1486	51.514	-0.147	200	113	48627332
3	51.5131	-0.1419	51.512	-0.139	500	235	48626504
4	51.5127	-0.1468	51.514	-0.147	100	145	48627332
5	51.5138	-0.1439	51.513	-0.140	400	284	55171098
6	51.5083	-0.1509	51.509	-0.151	100	78	48627465
7	51.50200	-0.0013	51.498	-0.010	300	749	53416308
8	51.48620	0.1105	51.490	0.120	300	782	54732850

An HTC-Android Smartphone was used to determine the client's Smartphone geographical location via the onboard GPS receiver (L_C in Table I). Simultaneously, the mobile phone tracking service "FollowUs" was used to determine the Smartphone location independently via cellular network triangulation on the authenticator side (L_A). The resultant localisation error L_E (distance between L_C, L_A), coming from the different accuracy of the two methods, is between 78 and 284 meters, with an average error of 160 meters for the genuine measurements. In addition to the location, the MO provides a proximity value (PV) to the authenticator (this PV expresses the confidence that the MO has in its own

localisation. I.e. the MO believes that the Smartphone is inside a circle around the provided location L_A with a radius equal to PV). The last column "Cell-ID" describes the basestation currently used in the communication process between client and MO and can be independently determined on both sides. On the client side, the cell-ID can be directly accessed via an Android system call. On the authenticator side, the cell-ID will be provided in addition to the location and PV by the MNO.

Correct interpretation of the received location-based challenge requires, on the client side, the corresponding de-obfuscation key "Key-$Loc_{De\text{-}Obf}$" to the obfuscation key "Key-Loc_{Obf}", used by the authenticator to create LocProg. To calculate the location-based key of the client's Smartphone at the client and bank side, for example, a location grid with size S and a unique key (K_1 to K_n) associated with every grid intersection is used (Figure 2).

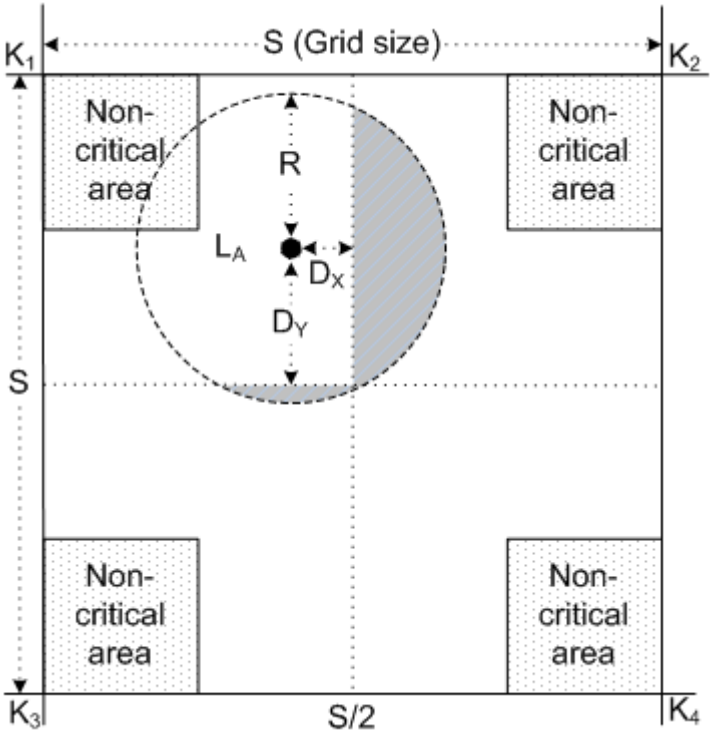

Figure 2. Location grid for key generation

The client and bank, for this example, use the key nearest to the determined (client) respectively requested (bank) location to (de-)obfuscate the program. This method always results in the same key on both sides, if a) the bank localises the client inside one of the "non-critical areas" around the intersections and b) the client is genuine and therefore less than the distance R away from the cellular based location L_A. Based on this example calculated error, shown in Table I, the circle around L_A with radius R describes the tolerance area for a client's position to be deemed as genuine.

In the experiments, a tolerance circle of R=300meters is used (this is because the FCC911 regulation which requires that network operators can locate Smartphones within an accuracy of 300 meters in 95% of all requests). So, if the MO returns a high confidence in his measurement, i.e. the PV is smaller than the default tolerance of 300 meters; the PV is used as the tolerance distance R. If L_A is outside the "non-critical areas", the genuine client might use the wrong key because the client is actually closer to another intersection as illustrated by the gray shaded areas in Figure 2.

To solve the problem of wrong key selection, the bank will accept challenge responses based on more than one key (e.g. K_1 and K_2 in Figure 2) if the result of the probability function (1) is smaller than a pre-defined threshold T. $D_{\{X,Y\}}$ expresses in Figure 2 and (1) the distance between the determined location and the middle of the grid area in the vertical (D_X), respectively horizontal (D_Y) direction.

$$P_{D_i} = \begin{cases} 1 & \text{if location is in non} - \text{critical area} \\ \dfrac{\mathrm{R} + D_i}{2R}, \mathrm{i} \in \{\mathrm{X}, \mathrm{Y}\} & \text{otherwise} \end{cases} \quad (1)$$

Note that accepting more than one key means that the area in which the genuine client needs to be is extended. In contrast, K_3 will not be accepted by the bank in Figure 2, as the probability (gray shaded area below the middle line) is smaller than the threshold T. If L_A is near to the centre of the grid, the challenge result based on all four keys (K_1 to K_4) will be accepted by the bank. In our trials, the (de-)obfuscation keys were calculated similar to the method described in [17]. The latitude / longitude values of the nearest grid intersection, PIN, KoP and cell-ID are combined as shown in Figure 3.

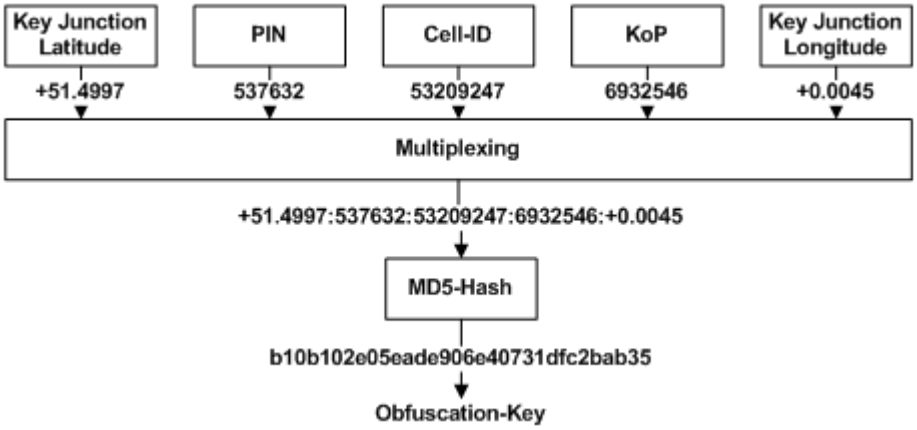

Figure 3. Key generation process for measurement 1

The combination is then hashed (e.g. MD5) and used as the key to (de-)obfuscate LocProg. Note that the cell-ID acts as a dynamic changing value and is used together with the client identifying PIN and KoP to increase the possible hash-function input value range and therefore strengthens the security and key-space of the LocAuth key.

Table 2 shows the generated client / bank keys for the 8 location measurements of Table 1. The used grid size is S=500 meters and the probability threshold is T = 0.95.

Table 2. Generated client and bank keys in trials.

	Prob P_X	Prob P_Y	Client Key	Bank Key_1	Bank Key_2	Bank Key_3	Bank Key_4	Key ok
1	**0.89**	**0.57**	**17675 ..**	**8170a ..**	**d1cc0 ..**	**17675 ..**	**da280 ..**	**True**
2	**0.69**	**0.89**	**fa008 ..**	**f66ad ..**	**fa008 ..**	**d1ebb ..**	**b7781 ..**	**True**
3	**0.72**	**0.71**	**e8a75 ..**	**44595 ..**	**e8a75 ..**	**65a63 ..**	**79ccb ..**	**True**
4	**0.89**	**1.00**	**f66ad ..**	**f66ad ..**	**fa008 ..**			**True**
5	**0.60**	**0.89**	**19baa ..**	**1375b ..**	**19baa ..**	**187a5 ..**	**ce489 ..**	**True**
6	**1.00**	**1.00**	**d6030 ..**	**d6030 ..**				**True**
7	**0.60**	**0.61**	**e6bdf ..**	**0cb34 ..**	**198d8 ..**	**0e007 ..**	**9599a ..**	**False**
8	**0.60**	**0.79**	**37f75 ..**	**c727a ..**	**31e87 ..**	**d9827 ..**	**b376e ..**	**False**

In the first row, both probability values (0.89 and 0.57) are below T (i.e. the client is near the grid cell centre) and the bank will accept the de-obfuscated responses based on all four surrounding keys. The client actually uses the key corresponding to Bank Key_3 (shaded) for the response. In row 6, the client location is determined by the bank in the non-

critical area (both probabilities are 1) and only one key is calculated. As the client uses the same key, the client's response will be considered as genuine. In the last two control data rows (7, 8); the response is not accepted by the bank as the client is too far away from the claimed location and therefore calculates a wrong key and a wrong LocProg result.

4.2 Location-Obfuscated Prototype Implementation

A prototype of LocAuth was developed and tested on the Android platform. The Dalvik virtual machine (DVM) which interprets all user applications running on Android Smartphones was changed to interpret LocAuth applications. The adapted DVM source code was re-compiled to test the viability of the prototype implementation. The integration of LocProg is done on the user application level once the client receives the challenge. Java annotations are used to distinguish between oBiometrics byte-code instructions (biometric-obfuscated) and instructions of LocProg (location-obfuscated) inside the DVM, e.g. "@Obfuscate(type=loc)". The DVM employs the correct de-obfuscation key (coming from fresh GPS location or biometric data) based on the annotation type during the interpretation. An example of a LocProg for Android Smartphones based on an integer equation is shown in Figure 4.

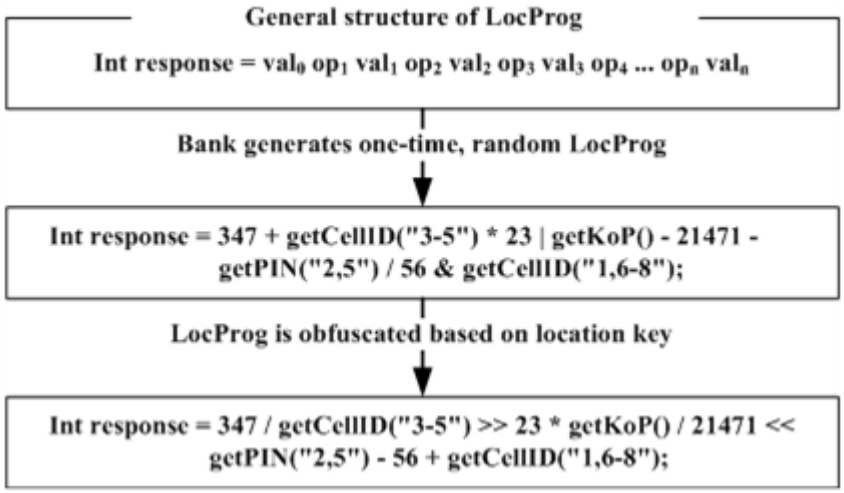

Figure 4. Example of challenge-program for Android Smartphones

The bank generates a random LocProg challenge-program and obfuscates the operators based on the location key. LocProg includes random elements (nonce, e.g. Integers), personal client data (getPIN(2, 5), to integrate the second and fifth digit of the PIN), the KoP token (getKoP()), and the dynamic shared value (getCellID(1, 6-8), which combines the digits 1, 6, 7 and 8 of the cell-ID into one integer).

As the operator substitutions must be within the same byte-code group (e.g. integer arithmetic: add-int, sub-int, …) to pass the byte-code verification of the DVM, all combinations for this group are expressed by n^{11}. Note that "n" corresponds to the number of operations in LocProg and the number of operators in the group is 11. For example, six operations in the above example would already lead to $1.77*10^6$ possible combinations.

4.3 Security and Performance Analysis

The following aspects were taken into consideration to assure that LocAuth security satisfies the application needs, yet remain practical with minimum overhead.

1. The tight combination of LocAuth with oBiometrics forces an attacker to breach all 5 authentication factors (biometrics, current location, PIN, KoP, and cell-ID) to gain access to the protected application. The experimental results of key generation for oBiometrics are discussed in detail in [5]. Integration of the current Smartphone location eliminates distance attacks because an attacker is not able to pretend to be at a different place. PIN and cell-ID are used as additional changing authentication factors in LocProg, thus increasing the possible key-space to make brute-force attacks more difficult. Furthermore, the use of two independent sources to compare the location of the client adds strength to the location authentication factor of LocAuth. This technique thus offer qualified level of security tailored to such Smartphone based applications yet offering advantage over current authentication mechanism.

2. To achieve a uniform distribution of the possible response values, and hence reduce the probability that the correct response could be guessed easily or discovered through a brute-force attack, LocProg must avoid certain combinations of operators and values. For example, LocProg should never have only multiplications with one value equal to 0. This combination would always lead to the final value of 0, regardless of the other values used.
3. LocProg is designed to always terminate correctly. If an incorrect key is entered, then a wrong result is generated. This "no-crash" property is important to eliminate locally performed "try-and-error" attacks because the attacker could conclude on its own that the wrong key was used and try other location keys until LocProg terminates correctly. To implement this, LocProg uses Integer arithmetic based on signed 32 bit values resulting in $4.29*10^9$ possible keys that make it infeasible for an attacker to guess the correct response value. If an arithmetic operation causes an overflow, no exception is raised and the program continues normally (i.e. without crash).
4. Execution of LocAuth introduces computational overhead. Each obfuscated instruction requires additional computing cycles to be de-obfuscated and then interpreted. However, as the number of total instructions in LocProg is small compared to the surrounding application with thousands of byte-code instructions, the overhead is negligible. In our trials, and in comparison with time consumed by running current authentication processes, we concluded that no performance degradation was noticeable.

5. CONCLUSION AND FUTURE WORK

In conclusion, this paper proposes the integration of a randomly generated, location-obfuscation, real-time challenge program into authentication based applications on mobile devices. LocAuth can be combined with oBiometrics software protection to securely authenticate a genuine client in all kinds of mobile security related transactions (mobile banking etc.) using enhanced mobile-devices like Smartphones. The integration of location as an additional, real-time, authentication factor, which is independently determined via two different sources on the client and authenticator sides, ensures a secure-live transaction and reduces the possibility of replay attacks. The challenge and calculated response using this technique can also be combined with other authentication factors (e.g. biometrics) to enhance the security of the authentication further.

The viability of the scheme is tested on the Android emulator. The Dalvik VM is adapted to interpret biometric- and location-obfuscated applications. The practicality of LocAuth is verified with actual geographical location measurements using Android-based Smartphones. A possible key generation approach to establish the same location-based (de-)obfuscation key on both the client and the authenticator sides completely independent from each other is described. An obfuscated challenge-program for Android mobile-devices based on integer arithmetic is also proposed. The implementation of the challenge-program is designed to "not-crash" when the incorrect authentication data is collected. This eliminates possible "try-and-error" attacks as an attacker can never be sure if the calculated result is correct.

Work on the location-obfuscated software protection scheme is ongoing. Future versions of the LocAuth scheme will incorporate other methods to independently determine the some location-based key. This shall reduce the possibility of wrong key selection on the client side and therefore enhance the accuracy and security of LocAuth. In addition, more sophisticated location-obfuscated challenge-programs will be developed. These programs will not only contain obfuscated operations based on the location, but also include other factors (e.g. time or velocity) in the obfuscated challenge / response program. Furthermore, these programs can protect the location privacy of the client using privacy preserving location verification (PPLV) [16]. PPLV allows establishing location based keys without revealing the actual physical location of the client to the authenticator. The authors are also investigating a dynamic scheme to ensure that the authentication sequence varies for every individual authentication. I.e. current authentication schemes typically follow a fixed sequence of steps at both sides.

Integration of a dynamically changing sequence of the authentication steps will prevent successful authentication of an attacker even if the personal data of the genuine client has been compromised. The proposal is to use an obfuscated sequence program to pass on the authentication sequence of the client's data used at that instance in time between the two parties (authenticator and client).

REFERENCES

[1] comScore, Inc., "Number of u.s. mobile financial account users surges 54 percent to 30 million in past year," Online, March 2011.

[2] Entrust, Inc., "Security's role in deploying transaction-enabled mobile applications," Online, Aug 2010.

[3] S. Li and A. Jain, Encyclopedia of Biometrics. Springer US, 2009.

[4] I. A. Lami, T. Kuseler, H. Al-Assam, and S. Jassim, "Locbiometrics: Mobile phone based multifactor biometric authentication with time and location assurance," in Proc. 18th Telecommunications Forum (IEEE TELFOR 2010). IEEE Telfor, Nov 2010.

[5] T. Kuseler, I. A. Lami, and H. Al-Assam, "obiometrics: A software protection scheme using biometric-based obfuscation," in 2011 African Conference on Software Engineering and Applied Computing (ACSEAC), Sep 2011.

[6] J. Caffery and G. Stuber, "Overview of radiolocation in cdma cellular systems," Communications Magazine, IEEE, vol. 36, pp. 38–45, 1998.

[7] C. Collberg, C. Thomborson, and D. Low, "A taxonomy of obfuscating transformations," Technical Report 148, Department of Computer Science, University of Auckland, Tech. Rep., July 1997.

[8] H. Chen, L. Yuan, X. Wu, B. Zang, B. Huang, and P. Yew, "Control flow obfuscation with information flow tracking," in Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture. ACM, 2009, pp. 391–400.

[9] A. Monden, A. Monsifrot, and C. Thomborson, "A framework for obfuscated interpretation," in ACSW Frontiers '04: Second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation. 2004, pp. 7–16.

[10] X. Zhang, F. He, and W. Zuo, "A java program tamper-proofing method," in SECTECH '08: International Conference on Security Technology. IEEE Computer Society, 2008, pp. 71–74.

[11] S. Saroiu and A. Wolman, "Enabling new mobile applications with location proofs," in Proceedings of the 10th workshop on Mobile Computing Systems and Applications. ACM, 2009, p. 3.

[12] V. Lenders, E. Koukoumidis, P. Zhang, and M. Martonosi, "Location-based trust for mobile user-generated content: applications, challenges and implementations," in Proceedings of the 9th workshop on Mobile computing systems and applications. ACM, 2008, pp. 60–64.

[13] W. Luo and U. Hengartner, "Veriplace: a privacy-aware location proof architecture," in Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems. ACM, 2010, pp. 23–32.

[14] L. Scott and D. Denning, "A location based encryption technique and some of its applications," in ION National Technical Meeting, vol. 2003, 2003, pp. 730–740.

[15] G. Yan, J. Lin, D. Rawat, and W. Yang, "A geographic location-based security mechanism for intelligent vehicular networks," Intelligent Computing and Information Science, pp. 693–698, 2011.

[16] T. Kuseler, H. Al-Assam, S. Jassim, and I. A. Lami, "Privacy preserving, real-time and location secured biometrics for mcommerce authentication," in Mobile Multimedia/Image Processing, Security, and Applications 2011, vol. 8063, SPIE, April 2011.

[17] G. Yan and S. Olariu, "An efficient geographic location-based security mechanism for vehicular adhoc networks," in 6th International Conference on Mobile Adhoc and Sensor Systems, MASS'09. IEEE, 2009, pp. 804–809.