Evaluation of real-time aspects of multiparty security on low-power mobile devices

Tobias Limmer*, Falko Dressler*, Ruben Gonzalez†

*Department of Computer Science, University of Erlangen
†Faculty of Engineering and Information Technology, Griffith University

Abstract. In this paper, we evaluated the performance of key exchange protocols on mobile devices. The wide usage of low power mobile phones causes mobile computing to spread rapidly and an increasing number of networked applications are developed for these devices. One important consideration in this scenario is data security. This is particularly important in secure groupware, collaborative or multiuser applications where simultaneous communication with multiple parties must be securely maintained. One key component of building secure data channels are computationally expensive key exchange protocols. Based on the benchmark of several asymmetric algorithms on mobile phones, we theoretically analyzed the speed of authenticated multiparty key agreement protocols with different designs and compared them with each other. The results were confirmed by a protocol benchmark in our testbed.

1 Introduction

In the last few years mobile computing has experienced an immense rise in popularity, mainly caused by the wide-spread use of mobile phones. The transmission of data may become the weak link as very few encryption methods are available for applications running on mobile phones. It allows third parties to easily intercept personal data during wireless transfer and during the routing through an insecure network such as the Internet.

This issue is even more important in multiuser applications like teleconferencing or shared white boards, where simultaneous communication between different parties must be securely maintained and all peers are realized on mobile devices. The network topology considered for this multiparty communication is a fully-meshed net: every member of the group can exchange data directly with each other. The usage of external devices introduces restrictions on the usability of multiparty protocols. So, that solution was avoided in this work.

One important aspect for this communication type is dynamic membership: Members should be able to join and leave the group without any restrictions. Separately managed point-to-point connections between hosts and the usage of already well-established secure 2-party protocols, e.g. SSL, are clearly suboptimal for this environment because group communication cannot take advantage of its unique features. Another major aspect is authentication to prevent man-in-the-middle attacks. All tested protocols had to support authentication and our preferred way of performing it was the usage of certificates with an underlying asymmetrical cryptographic algorithm, as those are most flexible in practical use.

The main limitation for the use of cryptography are the mobile phones' extremely limited processing resources. The market of mobile phones shows a wide variance of processing speeds, so we differentiated between low-cost models and a group of more expensive devices with more features and processing resources.

We benchmarked the asymmetric cryptographic systems RSA, ECC and XTR, as they provide means for authentication and many key exchange protocols use those algorithms. Based on these results, we analyzed the performance of multiparty key exchange protocols suitable for mobile phones. Previous work only considered the total number of exponentiations on each host and for each protocol round, those values only give a rough estimation of the time needed for the protocol execution.

This paper is structured in the following way: section 2 presents the limitations of mobile phones and reviews different cryptographic algorithms and secure communication protocols. We describe our analysis of secure group key exchange protocols and the validating benchmark in section 3, whose results are presented in section 4.

2 Overview and related work

2.1 Hardware and software environment

Processing capabilities of mobile phones are very limited as such devices are focused on providing long battery runtimes and small proportions. The current assortment can be roughly divided into two classes by regarding the processing power and operating system: The lower end class called feature phones which are cheap and form the majority of phones in use. An example is Nokia's series 40: They run on a Java based operating system, provide a 32 bit ARM-9 CPU with approximately 20 MHz and their total memory is 200 KBytes, but for Java applications only 70 KBytes are available. The higher end class called smart phones is more expensive and does not dominate the market. It is represented by Nokia's series 60 phones: based on a non-Java operating system called SymbianOS, they also support execution of Java applications and provide a faster CPU with approximately 120 MHz and more than 2 MBytes of memory. We explicitely excluded faster devices like PDAs.

The provided software environment on the target devices is the Java 2 platform, Micro Edition (J2ME) with the Connected Limited Device Configuration (CLDC) and Mobile Information Device Profile (MIDP). Our benchmarks were written in Java and were executed in this environment.

Several different wireless networks are available for mobile phones, each with different features for data transmission: The most popular system is the Global System for Mobile Communications (GSM), whose main focus is on voice transmission and provides a maximum possible data transfer rate of 14.4 kbit/s. The

General Packet Radio Service (GPRS) is an extension of the GSM standard and makes use of unused bandwidth for data transmission. Its theoretical speed limit is 170 kbit/s, but in practice usually 30-70 kbit/s are achieved. The successor of GSM is the Universal Mobile Telecommunication System (UMTS) with a theoretical maximum data rate of 1970 kbit/s, but currently only up to 320 kbit/s are supported.

2.2 Cryptographic algorithms

Cryptographic algorithms can be separated into two groups, depending on the number and type of keys used: Symmetric algorithms use the same key both for encryption and decryption and show good performance as they are based solely on simple and fast operations. Asymmetric algorithms use two keys, one for encryption (public key) and one for decryption (private key). The public key is usually made public, only the private key for decryption of data is kept secret. Rivest, Shamir and Adleman presented the first asymmetric cryptographic algorithm in 1978 [1]. It is one of the most used algorithms at the moment. Several works try to speed up calculations of RSA by outsourcing work to external servers [2,3].

A more efficient asymmetric algorithm than RSA is the elliptic curve algorithm proposed independently by Miller in 1985 [4] and Koblitz in 1987 [5]. ECC only needs small key sizes for an adequate security level, so less time is needed for calculations. Another interesting alternative to RSA was presented by Lenstra and Verheul in 2000 [6]: XTR. It is an abbreviation for 'Efficient and Compact Subgroup Trace Representation'. Similar to RSA, XTR is based on the discrete logarithm problem, but it uses traces to represent and calculate powers of elements of a finite field's subgroup. This enables the algorithm to perform faster than its predecessor RSA.

2.3 Key exchange in cryptographic protocols

Authenticated key establishment protocols perform two functions: They try to establish the authenticity of communicating peers and after successful execution, all parties are ensured that their peers are what they seem to be. The second function is the agreement of all participating hosts on a common shared key, which is usually needed for symmetric encryption of data.

The common process to establish and maintain secure symmetrical keys is realized in the following way: At first, a key encryption key (KEK) is exchanged by a key establishment protocol. This key is used to exchange group encryption keys (GEKs), which encrypt bulk data for a part of the session. After a certain time both hosts exchange a new GEK by using the KEK again. This process is called re-keying. We used key exchange protocols which rely on asymmetric cryptographic algorithms. Those are more flexible in usage, as they do not necessarily rely on predistributed information like symmetric algorithms.

Multiparty key exchange adds new challenges to its protocols compared to 2-party key exchange: Mutual authentication requires much more effort because of the greater number of hosts. Dynamic membership where hosts can join and leave the group dynamically also adds additional complexity. Multiparty protocols usually support different forms of key agreement: In an initial key agreement (IKA), the session key is established for the first time in a new group. When the membership changes and a host joins or leaves the group, auxiliary key agreement (AKA) is performed. AKA ensures, that former party members do not know the shared key after they left the group (forward secrecy) and that old shared keys are not retrievable for new party members (backward secrecy). Several secure protocols for multiparty applications have already been suggested [7,8,9]. They can be separated into contributory protocols and centralized protocols. Contributory protocols treat every party member identically and do not depend on a host which manages the security setup of the group. Almost none of introduced contributory group key exchange protocols support authentication of the members, and many early proposals were proven insecure [10]. Centralized protocols make use of one central host called group manager (GM), which controls the group's security and uses a 2-party key exchange protocol to distribute the keys.

3 Secure group key exchange

The performance of several group key agreement schemes was analyzed in this work. We assumed that all hosts are identical and perform all calculations equally fast. Algorithms based on symmetric cryptographic techniques are assumed to take no time for computation, as only little data is encrypted. The number of asymmetric operations was counted for each operation of the protocol. All calculations performed simultaneously on different machines were only counted as one calculation, as in that case no additional time was needed for the whole protocol execution. We also calculated the amount of message exchanges, as time needed for data transfer in wireless networks also impacts a protocol's performance. Transmission delays caused by the transfer of large amounts of data are not considered, as during key exchange only a small amount is transferred. So parallel message transfers and messages sent to multiple destinations are counted as a single message transfer. Message transfers which do not have any impact on the time needed for protocol execution were not counted. In the case that key exchange protocols were implemented, it may be possible that more message exchanges are needed than mentioned in this paper. We only considered the bare minimum amount for cryptographic security. We used the term no_e for the number of needed sequential exponentiations and no_c for the total number of sequential message transfers which delay protocol execution in a group of n hosts.

We analyzed the following operations of each protocol:

- *initial setup* – this operation is executed when a secure communication session is to be set up and all hosts agree on a common KEK
- *member join* – a new host U_{n+1} joins an already existing group
- *member leave* – a host leaves the group and the GEK will be changed

3.1 Contributory key agreement

Two contributory key agreement protocols were analyzed. The original protocols only agree on a KEK, so we included GEK generation and distribution by a randomly chosen member to reflect a more practical operation of the protocol. AKE1 was developed by Bresson in 2001 to 2003 [11]. It is one of the few contributory group key agreement protocols which are still considered secure. The hosts are arranged in a ring. All exchanged messages are signed using the hosts' long-term certificate and contain identifying strings. The signatures are verified by the receiving hosts. We assumed, that the protocol makes use of ECDSA which performs one EC multiplication for the signing operation and two multiplications to verify a signature.

TGDH was proposed by Kim et al. in 2000 [12] and was designed to be scalable for larger groups. Its primary structure is based on a fully balanced binary tree. The original protocol does not provide authentication, but it was included in the analysis because of its special structure which scales much better than the other contributory protocol AKE1. Authentication was artificially added to the protocol: every message exchanged between hosts was being signed by the sender and then the signature was verified by the receiving host. This was a theoretical measure to be able to compare the protocol with other authenticating schemes. It was not assumed that the protocol is secure against malicious attacks. Table 1 shows the results of the analysis of both protocols.

Protocol	*Operation*	no_e	no_c
AKE1	initial setup	$\frac{n(n-1)}{2} + 4n$	n+1
	member join	$2n + 7$	3
	member leave	$n + 2$	2
TGDH	initial setup	$5\lceil\log_2 n\rceil - 5$	$\lceil\log_2 n\rceil$
	member join	$5\lceil\log_2(n+1)\rceil - 5$	$\lceil\log_2(n+1)\rceil$
	member leave	$5\lceil\log_2(n+1)\rceil - 5$	$\lceil\log_2(n+1)\rceil$

Table 1. Results of theoretical analysis of contributory protocols

3.2 Centralized key distribution

In centralized key exchange, the group members do not perform any key agreement with each other, but only with the central GM. The process of centralized key distribution involves two steps:

1. The GM performs 2-party authenticated key exchange with all members in he group. As a result, the GM and each of the members share a KEK respectively.
2. The GM chooses randomly a common GEK and distributes it to all other members using the KEKs.

The key exchange protocol used in the initial exchange of the KEKs between GM and the other hosts needs to be as efficient as possible. Strangio and Popescu [13,14] performed an analysis of current 2-party authenticated key exchange protocols that use ECC. Almost all of the protocols require 2-3 asymmetrical operations per run. The most promising protocol is LLK which requires 2 operations for the key exchange per host. The time needed for key exchange increases linearly with the size of the group. New members are usually included in the group's encryption scheme by performing one 2-party key exchange and sending a new GEK to all members. Member leaves also require a new GEK to be sent to the remaining group members. Thus the total number of sequential multiplications and messages is constant for both member joins and leaves.

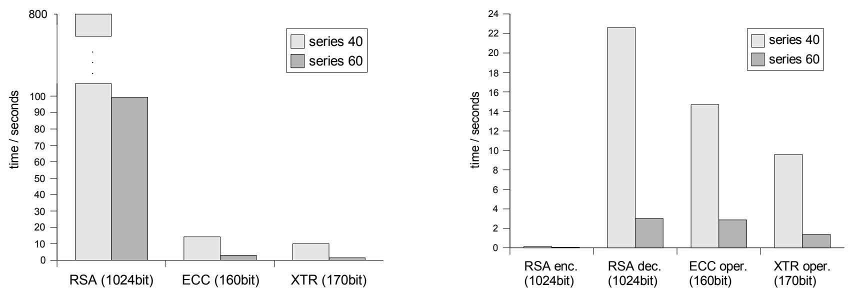
3.3 Benchmark of asymmetric cryptographic algorithms

The computationally most expensive calculations performed during key exchange are operations in an asymmetric cryptographic system. So we implemented a benchmark which tests the performance of RSA, ECC and XTR for the operations key generation, encryption and decryption with different key sizes. Relevant algorithms were taken from the libraries Bouncy Castle and Crypto++. The benchmark was written in Java as a MIDlet so that it was able to run in various mobile phones with MIDP support. A series 40 Nokia 6320 and a series 60 Nokia 6630 mobile phone were used for this test. The MIDlet was executed on both devices, so that the tests were performed identically. To obtain realistic results, the phones were connected to a phone network, but no other application was being executed at the same time. We used RSA with key size 1024 bit, ECC with 160 bit and XTR with 170 bit, which roughly provide equal security levels and are considered secure in this configuration at the moment.

4 Results and discussion

4.1 Asymmetric cryptographic algorithms

As described in section 3.3, we performed a benchmark of asymmetric cryptographic algorithms. The results are displayed in figure 1. RSA key generation showed a large speed variance during the tests, this was caused by the trial-and-error mechanism that tried to find an appropriate RSA key pair by random selection of numbers and a subsequent test for validity. Private and public key sizes were unbalanced in the used implementation of RSA. The public key was much smaller than the private key, so operations involving the public key (like encryption) took much less time than private key operations (like decryption). ECC operations showed more constant results, as no trial-and-error algorithm is used in any of its operations. The time for ECC and XTR encryption was twice the time for key generation and decryption, as two multiplications/exponentiations had to be performed for encryption instead of just one for key generation and decryption. Figure 1 shows the time needed for one operation in the cryptographic systems ECC and XTR.



(a) key generation　　(b) normal operations

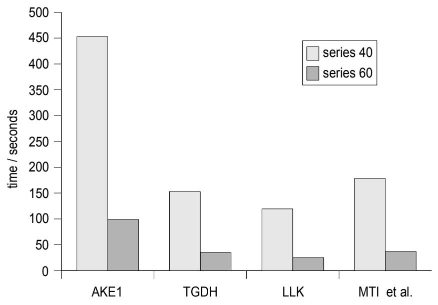
Fig. 1. Performance comparison of RSA, ECC and XTR on series 40 and 60 mobile phones

Key generation in the RSA cryptographic system took too long on mobile devices for practical use. Even on series 60 mobile phones, the process took in average 100 seconds. Although key generation had to be performed only once, the device was not be accessible at all during that time. Performance of ECC operations is a bit better compared to RSA decryption. This result was disappointing, as ECC is highly recommended in literature because of its vast performance advantages. A reason for this performance could have been the implementation of the Bouncy Castle Cryptographic API, which might have been inefficient. XTR operations showed slight speed advantages over ECC. The benchmark results suggested to use XTR, but as it was introduced only five years ago, minimal scientific effort has been invested into the algorithm.

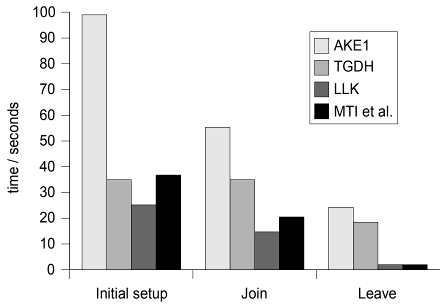
Overall, the use of ECC can be recommended for asymmetrical cryptographic operations. It was much faster in key generation than the current state-of-the-art cryptographic algorithm RSA and also showed performance improvements in decryption. Besides, much scientific work has been spent on ECC already and it is well recognized in the scientific community.

4.2 Key exchange protocols

We combined the results of the asymmetrical algorithms' performance test with the theoretical analysis of secure group protocols. Figure 2a compares the performance of initial setup in a set of series 40 and a set of series 60 mobile phones. The fastest initial setup of series 40 mobile phones was performed in 2 minutes, which is too much time for practical application. For further analysis, we only considered the faster group of series 60 phones. One operation in the elliptic curve was assumed to take 2.9 seconds, and a message transfer was assumed to take 2 seconds, as this is average in GSM data networks.The available network bandwidth was not included in our calculations. If many messages are sent simultaneously to many hosts, the bandwidth may become an important factor in the speed calculations. So the calculated values are lower bounds.



(a) initial setup on series 40 and 60 mobile phones

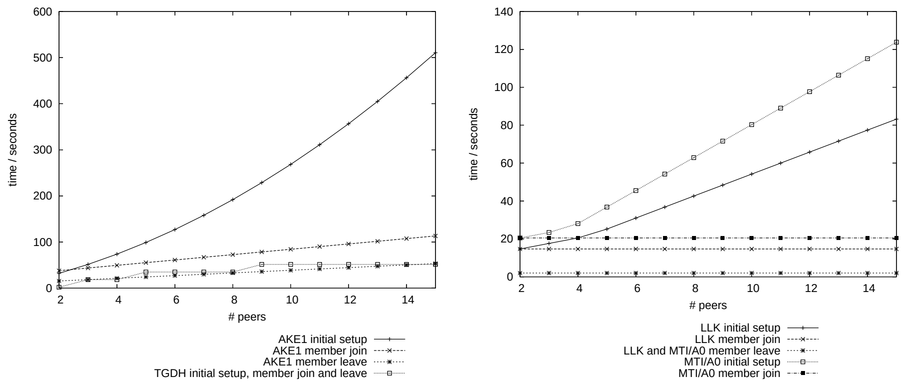


(b) different protocol operations on series 60 mobile phones

Fig. 2. Performance comparison of multiparty protocols with 5 group members

Figure 3 shows the results of the key exchange protocols' theoretical analysis. When setting up a new group, the complexity of AKE1 is $O(n^2)$. The cost of member joins and leaves is linear. The modified TGDH key exchange always performs the same operations during the group's initial setup, member join or leave. Its complexity is logarithmic to the group size, which results in much faster speeds than AKE1.The time needed to setup a group with the centralized group key distribution protocol based on LLK increases linearly to the group size. Up to a group size of 4 members, a small delay can be seen, as the GM needed to wait for some peers, before key exchange can finish. We included a slightly slower protocol called MTI/A0 in the diagram, which needs 3 asymmetrical operations to complete, to show how the speed of the 2-party key exchange influences the whole multiparty key exchange. Member joins need one 2-party key exchange and one message transfer to distribute the GEK. The protocols just perform one message transfer to distribute the GEK when a member leaves the group, so both LLK and MTI/A0 show the same speeds. We also performed a benchmark involving one series 60 and two series 40 mobile phones to validate the results of the theoretical analysis. The results confirmed our theoretical calculations.

In real environments, we do not expect more than ten members communicating with each other. A typical situation would be 5 members or less in a group. Figure 2b shows the times needed for each protocol and its operation. AKE1 is the slowest protocol in all three cases. The tree-based protocol TGDH is much faster in initial setup, but still needs twice the time for join operations and a member leave is more than ten times slower compared to the centralized protocols. MTI/A0 is slower than TGDH and LLK in the initial setup, but almost



(a) contributory protocols (b) centralized protocols

Fig. 3. Performance of key exchange protocols

twice as fast as TGDH in the join operation. Overall, LLK proved to have the best performance in all three operations.

Contributory protocols have security advantages over centralized protocols, but those are bought at an expensive price - especially AKE1's performance is much worse than the centralized protocol's performance. In each protocol operation, every member of the party needs to perform asymmetrical operations, which is not the case for centralized key exchange protocols. Those have some performance advantages through the use of symmetric encryption between group manager and members. Particularly leave operations are very efficient in centralized protocols: the GM chooses randomly a new GEK and the only notable delay is caused by the message transfer, whereas contributory protocols always have to perform computationally expensive asymmetrical operations. Unfortunately, centralized protocols are not well scalable, as the amount of computations on the GM is linear to the group size.

5 Conclusion

We analyzed several secure multiparty protocols for their feasibility to use them in groups only consisting of mobile phones, excluding the group of powerful PDA-like devices. Especially the case of smaller groups with 10 hosts or less was considered for groupware applications like teleconferencing. Special attention was given to the autonomy of the group and authentication of the peers.

Asymmetric cryptographic algorithms are required for the practical use of authentication. Our benchmarks showed, that ECC and XTR perform much better than RSA on the target platform. As XTR has been introduced only a few years before, we chose the more reliable alternative ECC for further analysis. Contributory protocols like AKE1 or TGDH always perform calculations on all hosts due to their decentralized nature. Centralized protocols using 2-party key exchanges like LLK or MTI/A0 have an advantage, especially in member join and leave operations, as only the group manager and one other host have to perform expensive calculations.

Our tests showed, that a centralized key distribution protocol using LLK is the best choice, but a time of 25 seconds for initial setup in a group of 5 devices on mobile phones of the faster generation is quick enough only for few applications. The current state of technology does not permit this type of communication yet. But the mobile phones' support for applications is improving: Processor speeds are increasing and more and more features are offered in their program environment. However, their processing power will still be a major concern for encryption in the years to come.

6 Bibliography

1. R. L. Rivest, A. Shamir, L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems.* Communications of the ACM archive, Volume 21, Issue 2, pp. 120-126, 1978
2. D. Boneh et al., *Generating RSA Keys on a Handheld Using an Untrusted Server.* RSA 2000
3. Yu Lei, Deren Chen, Zhongding Jiang, *Generating digital signatures on mobile devices.* 18th International Conference on Advanced Information Networking and Applications, AINA 2004. Volume 2, 29-31, pp. 532-535, 2004
4. V. S. Miller, *Use of Elliptic Curves in Cryptography.* Advances in Cryptology - CRYPTO, LNCS, Volume 218, pp. 417-426, 1985
5. N. Koblitz, *Elliptic Curve Cryptosystems.* Mathematics of Computation, Volume 48, pp. 203-209, 1987
6. Arjen K. Lenstra and Eric R. Verheul, *The XTR Public Key System.* Lecture Notes in Computer Science, Volume 1880, 2000
7. H. Harney and C. Muckenhirn, *RFC 2093: Group Key Manag. Protocol (GKMP) Specification.* Network Working Group, 1997
8. M. Steiner et al., *Diffie-Hellman Key Distribution Extended to Group Communication.* Proceedings of the 3rd ACM conference on Computer and communications security, pp. 31 - 37, 1996
9. M. Burmester and Y. Desmedt, *A Secure and Efficient Conference Key Distribution System.* In Proceedings of Eurocrypt 1994, pp. 275-286, 1994
10. M. Just and S. Vaudenay, *Authenticated Multi-Party Key Agreement.* Proceedings of the Int. Conference on the Theory and Applications of Cryptology and Inf. Security: Advances in Cryptology, pp. 36 - 49, 1996
11. E. Bresson et al., *Mutual Authentication and Group Key Agreement for Low-Power Mobile Devices.* 5th IEEE MWCN, 2003
12. Y. Kim et al., *Simple and Fault-Tolerant Key Agreement for Dynamic Collaborative Groups.* Proceedings of the 7th ACM conference on Computer and communications security, pp. 235 - 244, 2000
13. M.A. Strangio, *Efficient Diffie-Hellmann two-party key agreement protocols based on elliptic curves.* SAC '05, pp. 324-331, 2005
14. C. Popescu, *A Secure Key Agreement Protocol Using Elliptic Curves.* International Journal of Computers and Applications, Volume 27, 2005