

A Low Power Security Architecture for Mobile Commerce

Peter Langendoerfer⁺, Zoya Dyka⁺, Oliver Maye⁺ and Rolf Kraemer⁺

Abstract:

Mobile devices have limited resources in terms of computational power and energy. With the up-coming wireless Internet a lot of new applications that require security and privacy will be deployed. The exhaustive use of encryption mechanisms will drain down the battery within short time. In this article we present a security architecture that enables mobile devices to negotiate the cipher mechanisms with the infrastructure. Thus, the mobile devices are enabled to exploit the differences in the computational effort required for public and private key operations of RSA and elliptic curve cryptography (ECC). The benefit of this approach is that the mobile only has to execute RSA public key and EC private key operations. We present measurements for a given scenario, showing that by exploiting the differences in the computational effort of RSA and ECC, the computational burden on the mobile can be reduced by up to forty seven times.

Keywords: security architecture, elliptic curve cryptography, RSA, mobile devices, energy consumption

I. INTRODUCTION

Third generation mobile phones and Bluetooth or W-LAN enabled PDAs offer easy and efficient access to the Internet. The aspect of mobility and the actual location of the service user make it possible to address new types of tasks. We are convinced that a whole set of (probably only locally available) applications which adapt their content to the user's location or other usage context will significantly gain popularity among the next generation mobile applications. These services will be developed on top of location-aware middleware platforms. Such platforms provide functions such as profile handling, positioning and location handling, event services etc. An example for a location aware service is an "Airport Scout" that guides you to your gate. Such a service could charge a certain amount of money per meter of the guided distance.

Privacy and security are of crucial importance when it comes to mobile business. Here end-to-end security and privacy have to be provided on the application level. Encryption techniques can be used to ensure security and privacy. Public key encryption methods, which are definitely needed in this context e.g. for digital signatures, are computationally intensive. The problem that arises is

that all messages have to be encrypted in order to ensure privacy and that mobile devices have very limited resources in terms of calculation and battery power. The challenge here is to ensure privacy and security without a significant reduction of the up time of the mobile device.

In this paper we propose a security architecture that enables mobile devices to exploit the imbalance of computational effort of the private and public key operations of RSA and elliptic curve cryptography (ECC). In case of RSA, public key operations require a significantly smaller computational effort than private key operation whereas for ECC it is the other way around. The basic idea is to use an EC key pair on the mobile side and an RSA key pair on the infrastructure side. The benefit of this is that the mobile has to execute only RSA public key and ECC private key operations. We present measurements for a given scenario that show that by using this approach the computational burden on the mobile can be reduced by up to forty seven times.

Throughout this paper we use the reduction of the computational effort for a specific algorithm on a given device as an equivalent to the energy reduction.

This article is structured as follows. In the next section we describe the overall architecture of our security concept. Thereafter we describe the scenario we used in our performance evaluation. In section 4 we present our measurement results and discuss the potential performance gain of our approach. Then we draw conclusions on the applicability of our approach. The paper concludes with a short summary of the main results and an outlook on further research steps.

II. ASYMETRIC SECURITY ARCHITECTURE

A platform that supports m-commerce has at least one component that provides accounting and payment functionality. We think that location aware services will gain a significant popularity so that platforms deployed for m-commerce will be enhanced with components that provide location handling. The following types of data

⁺ IHP, Im Technologiepark 25, Frankfurt (Oder), Germany
langendoerfer@ihp-microelectronics.com

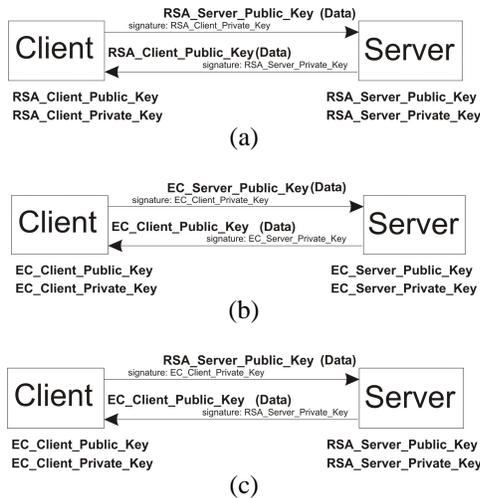


Figure 1: Application scheme of public key mechanisms. (a) pure RSA approach; (b) pure EC approach; (c) RSA on the infrastructure side and EC on the mobile side

must be exchanged between such an infrastructure (server side) and a mobile device:

- Current location of the mobile device
- Payment information (credit card, e-cash)
- Purchased goods (information, data)

If these data are not protected properly they can be used to develop detailed profiles of the subscribers. Since Internet subscribers are already highly concerned about their privacy this information has to be protected against eavesdropping and misuse [1]. Therefore all messages that are exchanged between the infrastructure and the mobile client as well as all messages that are exchanged between infrastructure servers should be encrypted. In addition, all these data have to be digitally signed in order to guarantee authenticity. Digital signatures are also needed to provide secure payment functionality. For signing messages public key cryptography is the best approach. This is due to the fact that with symmetric cryptography, each of the communication partners can easily forge a message. Thus, RSA and ECC or a combination of both should be used for signing messages. This can be handled in the following ways:

- The infrastructure and the mobile both use RSA for all purposes. The receiving side applies the public key of the sending side to verify the signature of the sending side as well as for encrypting its response. For generating its own signature it and for decrypting received data it uses its private key (see figure 1(a)).
- The infrastructure and the mobile device both use ECC for all purposes. The use of public and

private keys is equivalent to the RSA scenario (see figure 1(b)).

- The infrastructure uses its private RSA key to sign messages sent to the mobile. It encrypts these messages using the mobile client's public EC key. The mobile encrypts data using the public RSA key of the infrastructure and signs messages using its private EC key (see figure 1(c)).

The computational burden that arises from exhaustive use of public key cryptography will significantly decrease the up time of the mobile terminal. The minimal effort is achieved when a combination of RSA and ECC is used [2]. The reason for this phenomenon is that with RSA cryptography, public key operations are relatively easily to compute, whereas for ECC private key operations are calculated inexpensively.

In an environment in which the server side and the client do not know each other, a priori the use of encryption mechanisms has to be negotiated after the discovery phase. In order to minimize the encryption effort a symmetric key is exchanged in the suggested approach after the negotiation of the cipher algorithms.

A. Cipher Negotiation Protocol

In the following paragraphs we describe the initial communication set-up and how the cipher mechanisms are selected.

Mobile client and server side

During the initial communication set up the server side, and the mobile device have to agree about the encryption mechanisms that will be used. The server side broadcasts periodically its identity, its certificate and the set of encryption mechanisms it supports. When a mobile client receives this message it verifies the certificate of the server side using a set of stored public keys of trust centers which it knows. Then it responds with a message including its identifier, its public key and its preferred encryption algorithm for the uplink and the downlink, the selected symmetric cipher mechanism etc. Since the server side has unlimited resources in comparison to the mobile device, it accepts by default the proposal of the mobile device.

If the server side and the mobile device have no cipher mechanism in common they can agree to exchange messages without encrypting and signing of these messages. But in this case the server side as well as potential service providers have to agree explicitly.

In the next step the server side generates a pseudonym for the mobile client and a symmetric key for further communication. These data are encrypted and signed using the private key of the selected cipher mechanism of the server side and are then sent to the mobile client. The symmetric key is used to encrypt and decrypt all following messages. The pseudonym is used to identify the mobile

```

<?xml version = "1.0"?>
<!DOCTYPE cipher_mechanisms [ ...]
<UPLINK>
<PUBLICKEYMECHANISM>
  <ALGORITHM> RSA
  <PUBLICKEY> 123456789018745762362
</PUBLICKEYMECHANISM>
<PUBLICKEYMECHANISM>
  <ALGORITHM> B163 /*ECC*/
  <PUBLICKEY> F12654
</PUBLICKEYMECHANISM>
....
</UPLINK>
<DOWNLINK> .....
<DOWNLINK>
<SECRETKEYMECHANISM>
  <ALGORITHM>
</ SECRETKEYMECHANISM>
<SECUREHASH> MD5 </ SECUREHASH>
<SECUREHASH> SHA-1 </ SECUREHASH>
....
<NOSECURITYMECHANISM>

```

(a)

```

<?xml version = "1.0"?>
<!DOCTYPE cipher_mechanisms [ ...]
<UPLINK>
<PUBLICKEYMECHANISM>
  <ALGORITHM> B163 /*ECC*/
  <PUBLICKEY> F12654
</PUBLICKEYMECHANISM>
</UPLINK>
<DOWNLINK>
<PUBLICKEYMECHANISM>
  <ALGORITHM> RSA
  <PUBLICKEY>
    123456789018745762362
</PUBLICKEYMECHANISM>
</DOWNLINK>
<SECRETKEYMECHANISM>
  <ALGORITHM> DES
</SECRETKEYMECHANISM>
<SECUREHASH> MD5 </ SECUREHASH>

```

(b)

Figure 2: Messages of the cipher negotiation given in XML: (a) Security features supported by the infrastructure; (b) cipher mechanisms selected by the mobile device

and also to find the correct symmetric key for en- and decryption on the server side. All these messages are encrypted and digitally signed. The digital signature is verified before the message is decrypted. Thus, only valid messages are decrypted so that no useless operations are executed.

Mobile and service provider

The initial communication set-up between the mobile client and the service provider is very similar to those between mobile and infrastructure. If the mobile wants to use a certain service it sends a request with the service provider identity to the infrastructure. The server side responds with a message including the IP address of the service provider, the encryption mechanisms that the service provider supports and the public keys of the service provider. Then the mobile selects the cipher mechanisms for further communication and sends the corresponding message to the service provider. Thereafter the client can send requests to the service provider. All messages exchanged during the service usage are encrypted using the common symmetric key and signed with client's and service provider's private key respectively.

Since the infrastructure has provided the public key of the service provider, the mobile may skip the authentication of the service provider.

Message encoding

The messages exchanged during the negotiation of the cipher mechanisms are encoded using XML. Figure 2 (a) shows the message that is broadcasted by the server side;

the answer of the mobile is depicted in figure 2(b). The semantics of the message is defined in the document type definition. Thus, the mobile device can interpret the message even if there are cipher mechanisms that it does not know. So, it can select one algorithm per category according to its needs. Using XML encoded messages in the cipher mechanism negotiation phase has the benefit that there is no need to have standardized cipher chains like the ones used in TLS [3]. Therefore, extensions can be made for the server side and for the mobile side individually. During the connection set up the mobile and the server side create their own "cipher chain".

B. Key distribution

In this section we explain which of the system components (infrastructure, mobile client, and service provider) manage which keys.

Infrastructure

We assume an m-commerce infrastructure which covers a hot spot such as a train station or airport. This means it will consist of a set of access points, covering a certain part of that region, a set of servers responsible for a set of access points, and a single server that is responsible for the initial key exchange with new subscribers.

All messages exchanged between infrastructure components and between the infrastructure and mobile

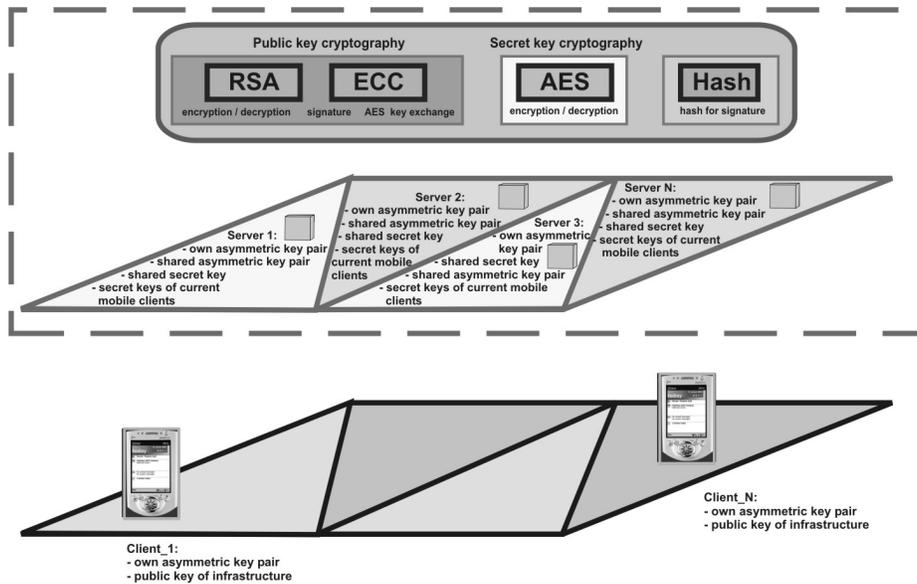


Figure 3: Structure of the infrastructure and key distribution within the infrastructure

clients must be encrypted. The messages exchanged between the server side components are encrypted using a symmetric encryption mechanism. For the initial communication set up with new clients, public key mechanisms are used. The corresponding keys are also used to sign messages that are sent to the mobile devices. Each infrastructure server has two public/private keys pairs: one individual and one common. The individual key pair is used to exchange the common symmetric key as well as to exchange the common public/private key pair. The latter is used for the communication with mobile clients. This approach has the following benefits:

- Any server can run the client subscription procedure.
- The mobile device must not authenticate new servers after a handover between two platform servers.
- The encryption mechanism, RSA or ECC do not have to be negotiated again after a handover.

After the initial communication set-up, a symmetric key is used to cipher data exchanged between the server side and the mobile. The subscribing server generates an individual public key for each mobile client. This key is forwarded to the client as well as to the server that covers the client's current location. It is also forwarded inside the infrastructure as part of the information exchanged during a handover.

Mobile device

Each mobile has its own public and private keys for asymmetric cipher mechanisms that it supports. In addition it stores the public key of the server side that fits to the selected cipher algorithms. It also has a key for the symmetric ciphering used for data communication with the server side. If the mobile has requested a service that is not provided by the infrastructure itself, the mobile manages also the public key of that service and a symmetric key for the data exchange with that service. The symmetric keys used for the communication with the infrastructure and the service are different.

Table 1. Number and type of public/private key operations on the client side.

Name of operations	Number of operations		
	Pure RSA	Pure ECC	Combination
Public Key of Infrastructure Encryption	RSA Public Key 2	EC Public Key 2	RSA Public Key 2
Private Key of Client Decryption	RSA Private Key 2	EC Private Key 2	EC Private Key 2
Private Key of Client signature generation	RSA Private Key 23	EC Private Key 23	EC Private Key 23
Public Key of Infrastructure signature verification	RSA Public Key 24	EC Public Key 24	RSA Public Key 24

Service Provider

The service provider stores the public key of each mobile that is currently served. Thus it is capable of verifying the payment tokens. In addition it manages one symmetric key that it uses to encrypt data sent to its current clients.

III. SCENARIO

The estimation of the time needed for encryption, generation, and verification of digital signatures is based on the following scenario. The mobile client and the server side negotiate the encryption mechanisms. After this initial phase the mobile requests a certain service. The server side provides the IP address and the set of supported cipher mechanisms and the public keys of the service provider. After receiving this information the mobile sends its selection of cipher mechanisms to the service provider. It responds with a pseudonym and a symmetric key for further communication. After receiving this information the mobile starts to create service requests. In our scenario we assume that it creates ten service requests which are served individually by the service provider. These messages contain a certain service request. The answer consists of the requested data, a service identifier, and the requested payment. The client responds with a payment token that represents the service identity and the requested amount of money. The service provider signs the payment token and sends it back as a receipt for the payment. In this scenario the following operations are executed on the client side:

- En-/decryption of the initial messages
- Signature verifications
- Signature generations
- Cipherng of the exchanged data with symmetric key.

For the performance evaluation we did not take into account the time for encrypting and decrypting data with the symmetric key. The messages exchanged are very small so that symmetric encryption is done extremely fast. More important is that the time needed for symmetric encryption operations is independent of the public key encryption features used. Thus, it has no influence on the performance gain of our approach.

IV. MEASUREMENTS

In order to verify whether the combination of RSA and ECC helps to reduce the computing effort on the client side we evaluated the computation time for the above sketched scenario. We investigated the following three cases:

- Client, infrastructure and service provider use RSA (pure RSA approach).
- Client, infrastructure and service provider use ECC (pure ECC approach).
- Client uses ECC for the up link and the infrastructure as well as the service provider applies RSA for the down link (combination approach).

The number and type of operations executed in our sample scenario are given in Table 1. The number of operations is independent of the public key mechanism applied.

We evaluated three different hardware configurations: a PC (PentiumIII with 450MHz), a HP Jornada (SH3 processor with 133MHz) and a Palm Pilot. For the ciphering we used RSA with a public exponent of 3, 17 and $2^{16}+1$ as well as B-163 and K-163 (recommended by NIST) that provide the same level of security as RSA-1024. We implemented the cipher mechanisms on a PentiumIII and on a HP Jornada. For our own implementation we used the MIRACL library from Shamus Software Ltd. (Ireland) [4], that provides operations in GF(p), GF(2^m) as well as modulo arithmetic implemented in C.

The execution time of the cipher operations is the only decisive part in our scenario. Therefore we did not measure the whole scenario including all lower layer operations and transfer through the network. Instead we measured each operation: signature generation, signature verification, en- and decryption for RSA and ECC 1000 times. Columns four and six of Table 2 present the average of these measurements for the PC and the Jornada implementation, respectively. We used these values to calculate the execution time for the cipher operations on the client side.

For the Palm Pilot device we used measurements presented in [5] to calculate the execution time for isolated cipher

Table 2. Time performance of cryptographic operations on the client side (Pentium III 450MHz and HP Jornada, MIRACL implementation); and Palm Pilot device, “Dragonball” 16MHz, calculation based on measurements presented in [5] in msec.

Parameters		Operation	time						
			measured on PC		measured on HP		calculation based on values from [5]		
			per operation	for scenario	per operation	for scenario	per operation	for scenario	
pure RSA	public exp: 3	RSA client private key signature generation	67.4	1698.0	18610	468838	36130	923.1	
		RSA server public key signature verification	0.5		138		729		
		RSA client private key decryption	67.4		18610		36284		
		RSA server public key encryption	0.5		138		1023		
	17	RSA client private key signature generation	67.4	1713.6	18610	473362	36130	931.65	
		RSA server public key signature verification	1.1		312		1058		
		RSA client private key decryption	67.4		18610		36284		
		RSA server public key encryption	1.1		312		1349		
	$2^{16}+1$	RSA client private key signature generation	67.4	1778.86	18610	491406	36130	965.87	
		RSA server public key signature verification	3.61		1006		2374		
		RSA client private key decryption	67.4		18610		36284		
		RSA server public key encryption	3.61		1006		2670		
pure ECC	EC: B-163	EC client private key signature generation	39.86	2417.5	266	16082	2230	198.21	
		EC server public key signature verification	53.88		357		5370		
		EC client private key decryption	48.78		331		3564		
		EC server public key encryption	55.02		367		5458		
	K-163	EC client private key signature generation	37.45	2431.21	253	15611	1793	128.63	
		EC server public key signature verification	57.08		354		3263		
		EC client private key decryption	41.35		284		1610		
		EC server public key encryption	58.62		364		2928		
combination approach	RSA public exp: 3	EC: B-163	EC client private key signature generation	39.86	1020.84	266	10368	2230	77.96
			RSA server public key signature verification	0.5		138		729	
			EC client private key decryption	48.78		331		3564	
			RSA server public key encryption	0.5		138		1023	
	17	EC: B-163	EC client private key signature generation	39.86	1036.44	266	14892	2230	86.51
			RSA server public key signature verification	1.1		312		1058	
			EC client private key decryption	48.78		331		3564	
			RSA server public key encryption	1.1		312		1349	
	$2^{16}+1$	EC: B-163	EC client private key signature generation	39.86	1101.70	266	32936	2230	120.73
			RSA server public key signature verification	3.61		1006		2374	
			EC client private key decryption	48.78		331		3564	
			RSA server public key encryption	3.61		1006		2670	
	RSA public exp: 3	EC: K-163	EC client private key signature generation	37.45	957.05	253	9975	1793	64.00
			RSA server public key signature verification	0.5		138		729	
			EC client private key decryption	41.35		284		1610	
			RSA server public key encryption	0.5		138		1023	
17	EC: K-163	EC client private key signature generation	37.45	972.65	253	14499	1793	72.55	
		RSA server public key signature verification	1.1		312		1058		
		EC client private key decryption	41.35		284		1610		
		RSA server public key encryption	1.1		312		1349		
$2^{16}+1$	EC: K-163	EC client private key signature generation	37.45	1037.91	253	32543	1793	106.78	
		RSA server public key signature verification	3.61		1006		2374		
		EC client private key decryption	41.35		284		1610		
		RSA server public key encryption	3.61		1006		2670		

operations. Then we summed them up in the same way as we did for the PC and Jornada in order to get the execution time for our scenario. These values are given in columns seven and eight of Table 2.

Columns four, six and eight of Table 2 show the execution time on the client side for our scenario for all RSA exponents (both elliptic curves as well as for the combination of the different RSA exponents and the

elliptic curves). The comparison of these values indicates clearly that RSA operations on the server and the mobile side need the longest execution time. On the PC and the Palm Pilot even EC on both side is outperformed by the combination of RSA public and EC private key operations. The factor that can be achieved by our approach depends strongly on the elliptic curve and the RSA exponent that are used for the operations. It reaches from 1.6 to 1.8 for the RSA part and from 2.2 up to 2.5 for the ECC part on

the PC. Table 3 shows these factors for all combinations of RSA exponents and EC curves we investigated. For the Palm Pilot the performance gain is even bigger. Here the factor ranges from 8 to 14.4 for the RSA part and for the ECC part it covers a range from 1.2 to 2.5 (see Table 4).

Table 3. Speed-up factor of using Combination approach on the Client side (Pentium III, 450MHz, MIRACL implementation) for proposed scenario.

Combination	RSA public exp:	pure RSA			pure ECC	
		3	17	$2^{16}+1$	EC	
					B-163	K-163
B-163	3	1.7	/	/	2.4	/
	17	/	1.7	/	2.3	/
	$2^{16}+1$	/	/	1.6	2.2	/
K-163	3	1.8	/	/	/	2.5
	17	/	1.8	/	/	2.5
	$2^{16}+1$	/	/	1.7	/	2.3

Table 4: Speed-up factor of using Combination approach on the Client side (Palm Pilot device, implemented [5]) for proposed scenario.

Combination	RSA public exp:	pure RSA			pure ECC	
		3	17	$2^{16}+1$	EC	
					B-163	K-163
B-163	3	11.8	/	/	2.5	/
	17	/	10.8	/	2.3	/
	$2^{16}+1$	/	/	8.0	1.6	/
K-163	3	14.4	/	/	/	2
	17	/	12.8	/	/	1.8
	$2^{16}+1$	/	/	9.0	/	1.2

Table 5: Speed-up factor of using Combination approach on the Client side (HP Jornada 540, 32MB, 133MHz, MIRACL implementation) for proposed scenario.

Combination	RSA public exp:	pure RSA			pure ECC	
		3	17	$2^{16}+1$	EC	
					B-163	K-163
B-163	3	45.2			1.6	
	17		31.8		1.1	
	$2^{16}+1$			14.9	0.5	
K-163	3	47.0				1.6
	17		32.6			1.1
	$2^{16}+1$			15.1		0.5

All combination of RSA and EC that we analyzed reduced the execution time for the PC and the Palm Pilot. When a HP Jornada is used as mobile client the situation changes. For the combination of K-163 and B-163 with RSA with exponent $2^{16}+1$ the performance decreases (see Table 5). Thus, for these combinations it is better to use ECC for the uplink and the downlink. But for all other combinations of RSA and ECC the performance is increased. The speed-up factor ranges from 31.8 for the RSA part and 1.1 for the ECC part up to 47 for the RSA part and 1.6 for the ECC part (see table 5).

We are not able explain in detail why the performance of the Jornada decreases for several combinations of RSA

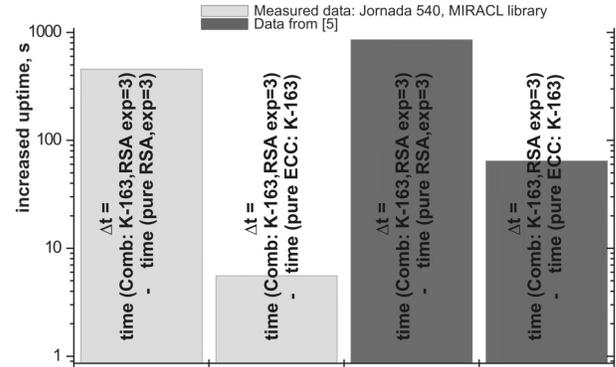


Figure 4: Increased uptime of the mobile devices when a combination of RSA and ECC is used

and EC. The implementation on the Jornada is exactly the same as we used on the PC. The problem seems to stem from the poor RSA implementation¹, which causes severe performance problems on the Jornada. Table 2 shows that the performance of RSA operation is 150 up to 300 times slower on the Jornada than they are on the PC. The ECC operations are slowed down only by a factor of seven.

The reduction of the computation time leads to a dramatically increased uptime for the mobile device. Depending on the mobile device and the cipher algorithms applied, the mobile device can be used up to 13 minutes longer (see Figure 4). This extension of the uptime was achieved for a very simple scenario in which only 26 public and 25 private key operations were executed on the client side. This indicates strongly that the features of our security architecture (cipher negotiation procedure, no new subscription procedure after internal handover etc.) offer suitable means to reduce the power consumption of mobile devices.

V. CONCLUSIONS

Communication set-up requires that all communication partners agree on the cipher mechanisms that will be used. Here two cases can be distinguished:

- mobile and a fixed infrastructure are communicating
- two mobiles are communicating

¹ e.g. we did not use Montgomery multiplication

In the former sections we discussed the first case. Our measurements show that in such a configuration, the mobile can benefit substantially from an asymmetric use of public key cryptography. Here the infrastructure will agree to use the computationally more intensive cipher mechanism. This is allowed since the infrastructure side has no limitations in terms of energy and computational power. The situation changes completely if two mobile devices try to set up secure communication. In this case, normally both communication partners will try to use ECC for their own sending path. If they agree to do so the computational burden is still smaller than with RSA on both sides. But the benefit in terms of energy and computational time-saving is much smaller than it is in the infrastructure case. There are some exceptions where one of the mobiles is willing to run the RSA part:

- If it is very eager to get a certain service from the other.
- If it will be recharged within a very short time.
- If it is equipped with specialized hardware, so that running RSA consumes less power than running ECC.

The results we measured on the HP Jornada indicate that the possible performance gain depends on the implementation. Therefore the selection preferences on the client side have to be chosen carefully. We recommend to measure the performance of all implemented cipher mechanisms in advance.

VI. SUMMARY AND OUTLOOK

We have presented a security architecture that reduces the computational burden on the mobile clients. The basic idea is to use two different types of public key mechanisms RSA and ECC, respectively. The computational effort needed for public key operations is relatively low for RSA but high for ECC. For private key operations it is the other way around. Thus, it is advantageous to use RSA for the downlink communication and ECC in the uplink communication. The measurements presented show that the computing time on the mobile side can be reduced up to a factor of 47 compared to a pure RSA approach. In comparison to a pure ECC approach the gain can go up to 2.5 if our approach is applied.

Our main goal is to reduce the energy consumption of the mobile device. Therefore we will focus on energy-efficient implementations of cipher mechanisms in our next research steps. The basic idea is to design hardware accelerators that can be used to support several cipher algorithms.

ACKNOWLEDGMENT

The work described here was partially funded by the German Government as part of the “Wireless Internet: Cellular” project under grant: 01AK047F

References:

- [1] Cranor L. F, 1999: Beyond Concern: Understanding Net Users' Attitudes About Online Privacy. In Ingo Vogelsang and Benjamin M. Compaine, eds. *The Internet Upheaval: Raising Questions, Seeking Answers in Communications Policy*. Cambridge, Massachusetts: The MIT Press, p. 47-70, 2000
- [2] Oliver Krone (ed.): *Jini & Friends @ Work: Towards Secured service access*. Eurescom Technical Information Project P1005, 2001 <http://www.eurescom.de>
- [3] IETF: The TLS Protocol Version 1.0, RFC2246, 1999, <http://www.ietf.org/rfc/rfc2246.txt>
- [4] Shamus Software Ltd. (Ireland): *The Multiprecision Integer and Rational Arithmetic C/C++ Library MIRACL (Version 4.7)*, <http://indigo.ie/~mscott/>
- [5] Michael Brown, Donny Cheung, Darrel Hankerson, Julio Lopez Hernandez, Michael Kirkup, Alfred Menezes: *PGP in Constrained Wireless Devices*. USENIX Association, Proceedings of the 9th USENIX Security Symposium Denver, Colorado, USA, August 14 –17, 2000 http://www.usenix.org/events/sec00/full_papers/brown/brown.pdf