TLS and Energy Consumption On a Mobile Device: A Measurement Study

Pedro Miranda, Matti Siekkinen
Aalto University, School of Science and Technology
Espoo, Finland
Email: firstname.lastname@tkk.fi

Heikki Waris†
Email: heikki.waris@gmail.com

Abstract— **We report results from a measurement study on the role of the most popular end-to-end security protocol Transport Layer Security (TLS) in the energy consumption of a mobile device. We measured energy consumed by TLS transactions between a Nokia N95 and several popular Web services over WLAN and 3G network interfaces. Our detailed analysis corroborates some earlier results but also reveals, contrary to earlier studies, that the transmission and I/O energy, both in the TLS handshake and the record protocol, far exceed the required computational energy by the actual cryptographic algorithms and that with transactions larger than 500KB, the energy required to transmit the actual data clearly outranks the TLS energy overhead. In addition, we note that the energy consumption varies remarkably between measured services.**

I. Introduction

In the last few years, mobile devices have evolved significantly in terms of power, throughput, and in terms of new functionalities, but they are still severely constrained by limited battery life-time. Secure communications are achieved by employing security protocols, which are based on cryptographic algorithms.

Executing certain cryptographic algorithms requires rather intensive computations. Therefore, their energy consumption on these battery powered devices is naturally a concern. In this paper, we study the energy consumption of Transport Layer Security (TLS), which is used to establish a secure communication channel between two end hosts and exchange data over that channel. TLS is a transport level protocol that uses asymmetric and symmetric encryption algorithms and hash algorithms in order to provide data secrecy, authentication of the communication parties, and data integrity for applications in a transparent manner.

We use a Symbian mobile device (Nokia N95) to establish TLS connections to different web services, such as electronic email or social networks, over both WLAN and 3G network interfaces and measure the energy consumption. TLS comprises two phases. First, a security association is created through a handshake protocol after which the actual data is transferred over an encrypted and integrity protected channel. We perform detailed analysis of the different steps involved in TLS transactions, compute the amount of energy overhead in a TLS transaction, and explain the major causes that determine this overhead.

Energy consumption of different cryptographic algorithms as well as the performance of TLS on PDAs and in computers have been studied earlier in, e.g. [1]–[3]. Among other things, they show that public key cryptography has the highest energy consumption, while hash algorithms have a little impact in the battery life-time. The key size chosen has a dramatic impact to the energy consumed in public key cryptography but not for symmetric algorithms. While some of our results corroborate those presented in these earlier studies, some of our conclusions concerning the overall TLS overhead differ significantly. In addition, we study real on-line services in a comparative manner and conduct experiments over both WLAN and 3G interfaces. Furthermore, we drill down into the individual steps involved in a transaction in order to pinpoint reasons for observed differences.

Our main findings are the following:

- The amount of energy consumed during the handshake phase differs a lot between different services. The main reasons turn out to be the length of the server certificate and RTT which are both related to transmission energy. Public key length seems to have only a marginal impact.
- Transactions over 3G access consume several times more energy than those over WLAN access. Contrary to earlier reported results, we found that the energy overhead of the TLS record protocol (encryption and hashing) is rather insignificant for WLAN and completely insignificant for 3G.
- For very small transactions (less than 10KB), the TLS overhead accounts for more than 60% (and up to 95% with DH) of the total energy consumed for both access types, while for transactions larger than 500KB the overhead is rather small if RSA is used in the handshake.

II. TLS and Energy Consumption

A. TLS overview

TLS [4] is a protocol that provides confidentiality, authentication, and data integrity over a channel between two machines. It is divided in two parts: *Handshake protocol and Record protocol.* The Handshake protocol allows the client and the server to authenticate each other and to negotiate the cryptographic algorithms and encryption keys needed to secure the channel. TLS packs the different cryptographic algorithms into cipher suites each of which specifies the server authentication algorithm, the key exchange algorithm, the bulk encryption algorithm and the message digest algorithm. Figure 1 shows the exchanged messages during a

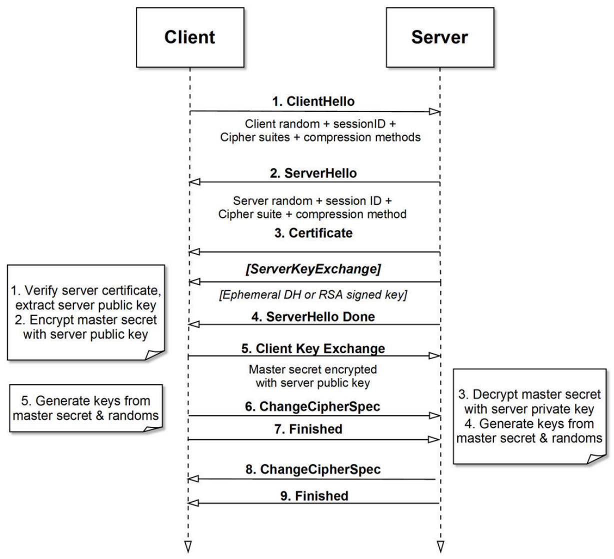
†Heikki Waris was with Nokia Research Center when this work was done.

TLS handshake using RSA or Diffie-Hellman(DH) as a key exchange protocol.

Unlike RSA, which can be used either for key exchange or for signing, DH can only be used for key agreement. As a consequence, Diffie-Hellman and RSA (or other protocol like DSS) have to be used together. The most common way to use DH is using ephemeral keys. The server generates a temporary DHE key, signs it with its RSA key, and transmits the signed key in the `ServerKeyExchange` message. In that case, the client will use that DH key for the key agreement. It is also possible to have a long-term DH key in which case the server will have a signed certificate containing the DHE key. The use of Diffie-Hellman as a protocol for the key agreement only adds one message from the server side, but it also adds different asymmetric operations that involve relatively heavy computations.

The most expensive part of the TLS handshake is the establishment of the *pre_master_secret*, which requires public key cryptography. The use of session resumption involves the reuse of a previous established session between the client and the server and thus only steps 1, 2, 7, 8 and 9 in Figure 1 are needed, thus avoiding the computationally expensive public key operations.

The Record protocol provides confidentiality and data integrity for the actual data transfer through the use of encryption and message digests.



Client
Server
1. ClientHello
Client random + sessionID + Cipher suites + compression methods
2. ServerHello
Server random + session ID + Cipher suite + compression method
3. Certificate
[ServerKeyExchange]
[Ephemeral DH or RSA signed key]
4. ServerHello Done
1. Verify server certificate, extract server public key 2. Encrypt master secret with server public key
5. Client Key Exchange
Master secret encrypted with server public key
5. Generate keys from master secret & randoms
6. ChangeCipherSpec
7. Finished
3. Decrypt master secret with server private key 4. Generate keys from master secret & randoms
8. ChangeCipherSpec
9. Finished

Fig. 1. Steps and exchanged data involved in a normal RSA handshake. The step within brackets is only used in the Diffie-Hellman handshake protocol

B. Where Does the Energy Go?

The use of the TLS protocol to secure a communication channel involves an overhead of computational work and exchanged data, that leads into a higher energy consumption. We can divide the energy consumed during an entire TLS transaction into cryptographic and non-cryptographic components. The former ones consist of the asymmetric operations, i.e. all the public key operations, symmetric operations, i.e. encryption and decryption of the bulk data, and hashing, i.e. message digest and digital signatures. The latter consists of the *transmission energy*, i.e. all the energy consumed while transferring data over TCP and maintain the network interfaces active. This energy is larger with TLS than without it due to increased data volume due to additional items such as certificates and message digests.

C. Energy Consumed by WLAN vs. 3G

The 3G interface operates in three different modes: *IDLE* mode is used in absence of network activity, *Dedicated Channel (DCH)* mode ensures the highest throughput with low delay transmission at the cost of a high energy consumption, and *Forward Access Channel (FACH)* mode shares the channel with other 3G devices and is used when there is little traffic to transmit, having roughly 50% of the power consumption of DCH mode. The transitions from DCH to FACH and from FACH to IDLE are controlled with operator set inactivity timers whose values usually are measured in several seconds. Because of this, there is so called *tail energy* which is spent in maintaining the high-power state after the completion of the data transmission, which, for example, can be as high as 60% of the total energy for a 50KB transfer [5]. We do not include this in our measurements.

The energy consumption of a WLAN interface depends on the following three factors: *scanning and association energy* is dissipated during the searching for an access point and connecting to one of them, *transmission energy* is required for the data transmission, and *maintenance energy* is used to keep the WLAN interface up. We do not include the scanning and association in our measurements. In addition, 802.11 includes a so called Power Saving Mode (PSM) which reduces energy consumption by enabling devices to go to a sleep mode and only regularly wake up to listen to a beacon which informs of new arriving data. We used off-the-shelf mobile device and access point in our measurements which support PSM and have it enabled by default.

III. Experiment Setup

We chose to use Advanced Encryption Standard (AES) because it is one of the most used algorithms and defined as a cryptographic standard by the NIST [6]. Furthermore, previous studies of symmetric algorithms [1] have shown that AES has good performance with competitive energy costs, and it is a sufficiently recent and secure standard that has relevance in actual implementations and products on the market. Thus, the AES cipher suites extending TLS v1.0, defined in [7] have been used, cf. Table II. From the 12 cipher suites supporting AES in the TLS, we discarded some for the following reasons: We wanted to study the differences between RSA and DHE. The use of Anonymous Diffie-Hellman (ADH) provides confidentiality but no authentication, which makes man-in-the-middle attacks possible. The use of the Digital Signature Standard (DSS) results in a slower performance [4] if it is compared to RSA. OpenSSL only implements ephemeral Diffie-Hellman (DHE), but not Diffie-Hellman (DH).

TABLE I

FULL PROFILE OF THE REMOTE SERVICES ANALYZED

Service	Google	Facebook	SSL.Facebook	M.Facebook	Verisign	Ovi
Protocols available	SSLv3, TLSv1.0	SSLv3, TLS1.0	SSLv3, TLS1.0	SSLv3, TLS1.0	SSLv3, TLSv1.0	SSLv3, TLS1.0
Server Implementation	Apache mod_SSL	Apache mod_NSS	Apache mod_NSS	Apache mod_NSS	Apache mod_SSL	IIS 7.5
Default Cipher Suite	AES256-SHA	RC4-MD5	RC4-MD5	RC4-MD5	DHE-RSA-AES256-SHA	DHE-RSA-AES256-SHA
Server Certificate length	1625 bytes	4553 bytes	4642 bytes	836 bytes	4519 bytes	1738 bytes
Certificate chain length	2	3	3	1	3	1
Public server-key	1024 bits	1024 bits	2048 bits	1024 bits	2048 bits	1024 bits
Session resumption	YES	NO	NO	NO	YES	YES
Ephemeral Diffie Hellman	NO	NO	YES	YES	YES	YES

TABLE II

LIST OF CIPHER SUITES SELECTED FOR THE EXPERIMENTS

Cipher Suite	Auth	Key Exchange	Encryption	Digest
RSA-AES-128-SHA	RSA	RSA	AES-128-CBC	SHA1
RSA-AES-256-SHA	RSA	RSA	AES-256-CBC	SHA1
DHE-RSA-AES-128-SHA	RSA	DHE	AES-128-CBC	SHA1
DHE-RSA-AES-256-SHA	RSA	DHE	AES-256-CBC	SHA1

The measurement setup consists of a client run on Nokia N95 phone that connects to a public server through a dedicated WLAN (802.11 b/g) access point or 3G (WCDMA with HSDPA) access network. The client program was developed using OpenSSL [8]. N95 does not have hardware acceleration for crypto operations contrary to some newer phones models.

We measured energy by using the Nokia Energy Profiler (NEP: http://www.forum.nokia.com/energyprofiler). In order to get as accurate and unbiased results as possible about the energy consumed by the TLS transaction, the client software automatically triggers the measurements with NEP through the command line interface. The NEP GUI is not used to minimize the power draw by the display. The TLS handshake process involves operations that take execution times in order of magnitude of milliseconds. To guarantee accurate results in the experiments, many repetitions of the different processes and experiments have been done. This is also important due to the fact that the Nokia Energy Profiler has a maximum sampling rate of only 4Hz, while the smallest TLS transactions can be executed very quickly. Thus, in certain cases like using AES[128,256]-SHA1, the required number of repetitions can be up to 300 from which the average energy consumption is computed.

IV. ANALYSIS OF THE HANDSHAKE ENERGY

We analyzed the energy consumption several different services. Table I shows the profiles for each of the studied services. The server implementation was obtained using SSLAudit (http://www.g-sec.lu/products.html). We focus first only on the handshake part of TLS.

A. *Energy consumption using WLAN and 3G*

Figures 2(a) and 2(b) show the energy consumption of the handshake phase only when using WLAN and 3G interfaces. We observe that the use of DH increases energy consumption a lot. The main reason is that DH is computationally much more intensive, but this fact also causes the transaction to last longer which increases the amount of energy consumed by the network interface.

The second thing we notice immediately is that a handshake performed over 3G access consumes in each case at least twice the amount of energy and in some cases up to four times the energy that is consumed over WLAN. The results in [5] report also such a very large difference for very short transfers between 3G and WLAN. However, contrary to that study, we did not include the 3G tail energy (since transfer phase starts right after handshake) which for very short transfers constitutes a vast majority of the energy consumed. For this reason, our results are surprising.

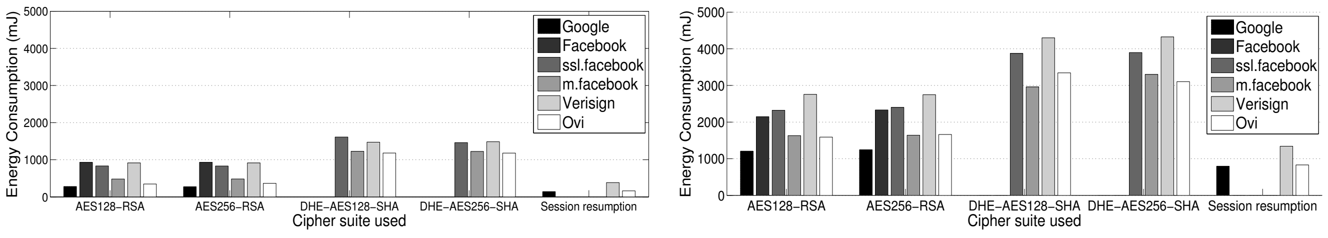
Perhaps the most interesting observation is the striking difference in energy consumption between the tested services. TLS handshake with Google and Ovi servers consume the least energy, while with Facebook and Verisign the energy consumption is more than doubled in case of RSA over WLAN and is clearly higher over 3G. These results and the large difference between WLAN and 3G access require more detailed investigation which we perform in the next section.

The use of session resumption greatly reduces the energy consumption by eliminating the expensive steps. Resumption was only supported by Google, VeriSign, and Ovi servers.

B. *Stepwise Analysis*

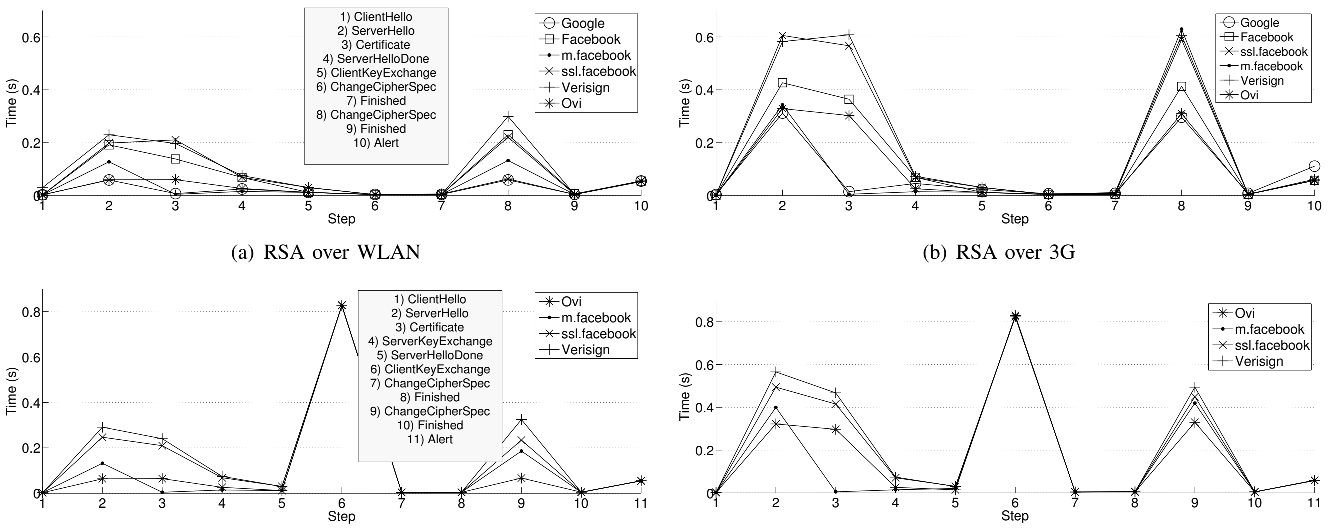
In order to understand which of the individual steps (see Figure 1) are responsible for the differences observed in the overall energy consumption, we analyze the execution time of each step. Figure 3 shows the results. The values are averages over 5 consecutive experiments.

We observe that the biggest differences in terms of execution time happen during the `ServerHello`, `Certificate`, and `ChangeCipherSpec` steps. `ServerHello` command sent by server in response to `ClientHello` includes various information such as the TLS version and the cipher suite to be used and random data for generating the secret-key. In `Certificate` step, the server sends its certificate including the public key. Optionally, it also includes the chain of certificates beginning with the Certificate Authority. In both steps, no cryptographic operations are involved by the client. Only after the certificate has been received, the client authenticates the server by checking the signature in the certificate, which is not that expensive operation with RSA [1]. Therefore, the delay differences are due to transmission using TCP over different RTTs. RTTs to Google and Ovi



(a) WLAN (b) 3G

Fig. 2. Energy consumption per connection in different remote services using WLAN and 3G.



(a) RSA over WLAN (b) RSA over 3G

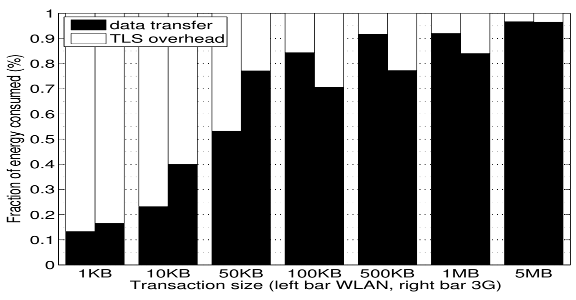
(c) DHE over WLAN (d) DHE over 3G

Fig. 3. Execution time of the different steps of a handshake.

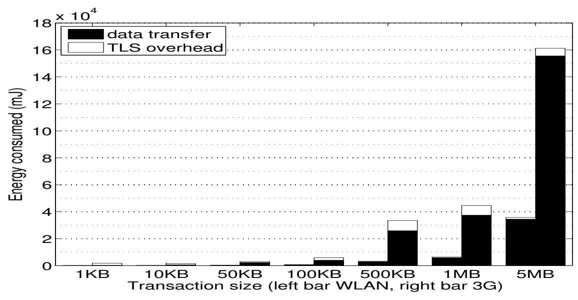
are especially short (< 50ms) while Verisign and Facebook servers seem to be located overseas in the USA. There is one RTT included in step 2 and, in addition, another RTT in step 3 for Verisign and Facebook due to large certificates (see Table I). Indeed, TCP applies slow start for increasing the congestion window size and even with initial window size of 2 packets, a 4KB certificate does not fit into these two packets together with the `ServerHello` message. Thus, the server needs to wait another RTT for ACKs to arrive from the client before transmitting the rest of the certificate data. Similarly a RTT is included in step 8 although very little data is actually sent by the server. In addition, the duration of that step is slightly longer for some services compared to step 2 because the server decrypts client encrypted message using private key which is a rather heavy operation with RSA (compared to the public key operations performed by client). This operation can add a small extra delay especially if the server is loaded.

The above described delays have a major impact on the energy consumption since the network interfaces continue to consume energy even if no data is received. The impact is clearly larger for 3G because in the network interface stays in the highest power state (DCH) constantly: the operator set inactivity timers do not expire in between sending and receiving messages. WLAN interface is able to transition in between messages into idle or even sleep mode which consume significantly less energy [9]. Indeed, to give some reference numbers, a single message exchange without any computation (e.g. TCP handshake) consumes between 400-500mJ over 3G with 300ms RTT (power consumption between 1.3-1.5W). In comparison, the two cryptographic public key operations (server authentication after step 3 and encryption for step 5) that client needs to perform consume in the order of tens of mJ [1]. Some of the servers use 1024 bit and some 2048 public keys and the respective energy consumptions for cryptographic operations are roughly 10mJ and 50mJ [10]. So, in any case the numbers are rather negligible when compared to the communication overhead, which can also be seen by comparing the energy consumed by Facebook and SSL.Facebook.

The certification chain length can also vary and it determines, if implemented correctly, how many signature verifications the client needs to perform in a sequence. We list that separately in Table I although the certificate length in bytes already indicates the difference. As mentioned before, this operation is not that heavy and the differences in chain length show as small variation in the execution time of `ServerHelloDone` step between servers in Figure 3 and



(a) Relative portion of overhead



(b) Total amount of energy consumed

Fig. 4. Measured energy overhead of TLS usage for WLAN and 3G.

the impact to energy consumption seems to be negligible compared to other factors.

Note that client authentication was not used since none of the services allowed it. That procedure would require the client to generate a signature and send a certificate, thus consuming much more energy with RSA.

Using Ephemeral Diffie-Hellman as a key agreement protocol improves the security in the TLS handshake, but it comes with a cost in terms of delay and energy consumption. `ClientKeyExchange` is the computationally hardest step. This step factors the time it takes for the client to perform the necessary computations. These results also agree with those reported in [2]. Google and Facebook did not support the Diffie-Hellman protocol.

V. Total Energy Overhead of TLS

In order to measure the communication data overhead imposed by TLS, we setup a server locally and performed uploads[1] of varying file sizes with and without TLS, i.e. comparing TLS transaction to plain TCP transfer. We used the RSA-AES128-SHA cipher suite with a 1024-bit server public key in all the experiments. RTT over WLAN was just a few milliseconds as the server and client were in the same local network and RTT over 3G was around 200ms. For the WLAN experiments, we imposed a rate limit of 200KB/s using Trickle[2] because we observed that without any artificial bandwidth limitation, the results were biased by the limited CPU capacity of the mobile device: The throughput was slowed down by the rate at which it could perform per-packet cryptographic operations (encryption and hashing) when the transaction size exceeded 50KB, which caused the throughput achieved to be notably lower than the throughput achieved during a plain data transfer. We know from earlier work that the energy consumption of TCP transfer over WLAN is strongly dependent on throughput [9] in such a way that the higher the throughput, the less energy is consumed per bit transferred. With a rate limiter, this problem was solved.

[1]Download results should be similar for WLAN because transmit and receive states consume roughly the same amount of power [9]. For 3G downloading is somewhat cheaper energy wise than uploading [5].

[2]http://monkey.org/ marius/pages/?page=trickle

A. Measured Total Overhead

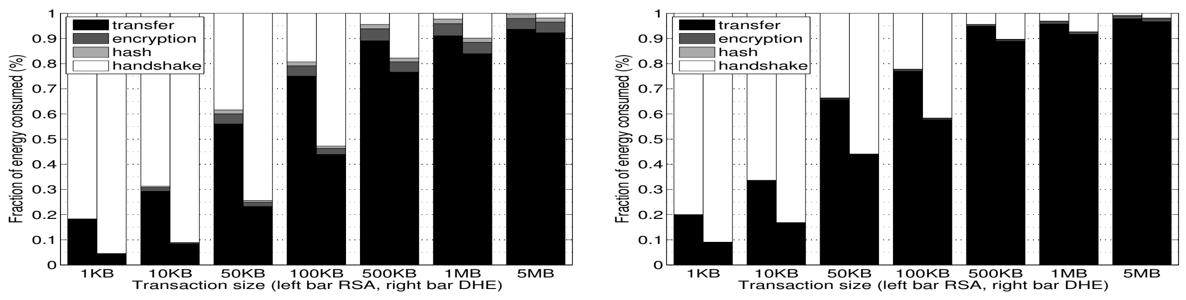
Figure 4(a) shows the measured overhead in relative values computed simply by comparing the total energy consumed by TLS transaction to that of plain TCP transfer. As expected, the energy overhead decreases as the amount of data to transfer increases because the energy required by the record protocol, including transfer energy, amortizes the energy used for the handshake. We also note that when the transaction size increases the overhead decreases more rapidly when transmitting over WLAN than over 3G, which is mostly due to the relatively higher energy consumption of handshake over 3G. We should point out that while the measurements over WLAN were rather stable, the measurements over 3G were not. In fact, the results fluctuated quite a lot due to the much more unstable nature of the 3G communication channel (jitter and bandwidth variation), and the coefficient of variation for the results varied from 0.07 (5MB transaction) to as high as 0.65 (1KB transaction).

We plot the total energy consumed during the experiments in Figure 4(a) which reveals that 3G consumes several times more energy than WLAN. In fact, just the TLS energy overhead for 3G is as much as the energy consumed by the entire transaction over WLAN up to 1MB size.

Our results give quite a different picture of the overhead compared to those reported in [1]. Indeed, this earlier work computes the TLS overhead to be as high as 55% in the case of 1MB transaction (over WLAN), while in our measurements this overhead is less than 10%. We believe that we are able to explain this difference as we dig deeper into the different parts that contribute to the overhead in the following section.

B. Computed Overhead Breakdown

Based on detailed measurements on the mobile device, we computed also the share of each cryptographic operation involved in the TLS overhead. As in [1], we measured the energy consumption of the individual operations and the results are in Table III. The measurements were performed separately with the same N95 device by running a custom program that performs a defined number of simple operations and measuring the energy consumed with NEP. We rounded the values to one decimal as the standard deviation over five consecutive experiments varied from 0.02 to 0.06. While the



(a) WLAN (b) 3G

Fig. 5. Break down of TLS energy consumption into cryptographic and non cryptographic components.

other results are in the same order of magnitude as reported in earlier work, the most important to note are the results from the plain I/O experiments. These results suggest that the symmetric key and hash operations are in fact very cheap energy wise: encrypting with AES-128 and hashing with SHA1 consume only $0.7\mu J/B$ and $0.1\mu J/B$, respectively, when removing the I/O part, which means that it is the reading and/or writing the source plain or cipher text file that consumes the most energy in these operations and not the execution of the actual crypto algorithm. It appears that this I/O overhead was not taken into account in the previous studies.

TABLE III
ENERGY CONSUMPTION OF INDIVIDUAL TLS OPERATIONS.

Operation	Energy ($\mu J/B$)
Encrypt with AES-256 (r/w phone memory)	2.0
Decrypt with AES-256 (r/w phone memory)	2.0
AES-256 vs. AES-128	0.3
Hash with SHA1 (read phone memory)	0.7
Read only (phone memory)	0.6
Read & write (phone memory)	1.3
Read only (memory card)	1.2
Read & write (memory card)	2.9

We computed the break down of the TLS energy consumption into the different components using the base data in the table and separately performed measurement values for the handshake with the local server. Measurement results from non-TLS experiments presented in the the previous section were used as the transfer energy[3]. Figures 5(a) and 5(b) show the results for the cases of WLAN and 3G, respectively. It is easy to notice that while encryption and hashing play a small part when communicating over WLAN, they are completely negligible in the 3G scenario. On the other hand, the handshake is an important factor to take into account but, as we showed earlier, this is mostly due to communication overhead during the handshake in the case of RSA but not entirely in the case of DHE. The computed overhead does not match perfectly with the measured one (Figure 4(a)) in the case of 3G due to varying measurement results, but in the WLAN case the results are very close to each other.

[3]The true transfer energy is slightly higher because TLS record protocol causes some overhead also in the amount of data to transfer but we measured this overhead to be only about 3-4%.

VI. DISCUSSION

While HTTP/1.1 specification states that "a single-user client SHOULD NOT maintain more than 2 connections with any server or proxy" [11], browsers nowadays routinely break this part of the specification and establish many more connections [12]. This has a significant impact to a mobile client's energy overhead when using TLS especially when session resumption is not allowed by the server, which somewhat surprisingly in our experiments turned out to be the case with Facebook, for instance.

Other opportunities for optimization, especially for short transfers where the TLS handshake plays an important part, are reducing the size of the server certificate and avoiding the use of Diffie-Hellman key exchange instead of RSA unless absolutely necessary. Otherwise, reducing, for instance, the length of the server's public key has little impact.

As we explained earlier, we used only supported cipher suites and, therefore, did not analyze suites based on Elliptic Curve Cryptography (ECC). Earlier results show that compared to RSA, while digital signing using ECC costs less energy, verifying a signature can consume up to ten times more energy [1]. This is important since in a typical transaction with a web server client verification (i.e. signing by client) is not used but the client may need to perform multiple signature verifications during a handshake.

The N95 does not include hardware acceleration for cryptographic operations unlike some newer devices. For example, the Nokia E-series devices and some iPhone provide this support for at least encryption using AES. We did not try to measure the impact of such optimizations on energy consumption, but we expect the impact to be small, given that according to our results the contribution of symmetric crypto operations to the overall energy consumption is very small.

Perhaps the most important take away is that, contrary to earlier studies, our results reveal that once the TLS transaction size exceeds 500KB, the overhead becomes much less important regardless of whether WLAN or 3G is used as the access network. In addition, the overhead of encryption and integrity protection by the record protocol is rather meaningless. Thus, optimizing the choice of symmetric crypto or hash algorithms makes little sense from the energy consumption perspective, except if the transmission rate is very high in which case more efficient crypto algorithms may prevent them to become a bottleneck and to limit the throughput and lower the energy efficiency.

VII. Conclusions

We presented in this paper a measurement study of the energy consumption of TLS protocol on a mobile device. Handshake energy consumption varies considerably between the measured services. Contrary to results reported in earlier studies, we observed also that the transmission and I/O energy, both in the TLS handshake and the record protocol, far exceeds the required computational energy by the actual cryptographic algorithms and that with transactions larger than 500KB, the energy required to transmit the actual data clearly overweighs the TLS energy overhead.

References

[1] Nachiketh R. Potlapally, Srivaths Ravi, Anand Raghunathan, and Niraj K. Jha, “A study of the energy consumption characteristics of cryptographic algorithms and security protocols,” *IEEE Transactions on Mobile Computing*, vol. 5, no. 2, pp. 128–143, 2006.

[2] Youngsang Shin, M. Gupta, and S. Myers, “A study of the performance of ssl on pdas,” in *INFOCOM Workshops 2009, IEEE*, apr. 2009.

[3] George Apostolopoulos, Vinod Peris, and Debanjan Saha, “Transport layer security: How much does it really cost,” in *In Proceedings of the IEEE INFOCOM*, 1999.

[4] Eric Rescorla, *SSL and TLS: Designing and building Secure Systems*, Addison-Wesley, 2001.

[5] Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani, “Energy consumption in mobile phones: a measurement study and implications for network applications,” in *Proceedings of IMC 2009.*, 2009.

[6] National Institute of Standards and Technology, *Specification for the ADVANCED ENCRYPTION STANDARD (AES)*, 2001, Available at http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[7] P. Chown, “Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS),” RFC 3268 (Proposed Standard), 2002, Obsoleted by RFC 5246.

[8] *OpenSSL project*, 2010, Available at http://www.openssl.org.

[9] Yu Xiao, Petri Savolainen, Arto Karppanen, Matti Siekkinen, and Antti Ylä-Jääski, “Practical power modeling of data transmission over 802.11g for wireless applications,” in *Proceedings of e-Energy 2010.*, 2010.

[10] A.S. Wander, N. Gura, H. Eberle, V. Gupta, and S.C. Shantz, “Energy analysis of public-key cryptography for wireless sensor networks,” in *Proceedings of PerCom 2005.*, mar. 2005.

[11] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, “Hypertext Transfer Protocol – HTTP/1.1,” RFC 2616 (Draft Standard), June 1999, Updated by RFC 2817.

[12] Nandita Dukkipati, Tiziana Refice, Yuchung Cheng, Jerry Chu, Tom Herbert, Amit Agarwal, Arvind Jain, and Natalia Sutin, “An argument for increasing tcp’s initial congestion window,” *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 3, 2010.