Are Wearables Ready for HTTPS? On the Potential of Direct Secure Communication on Wearables

Harini Kolamunna,[1,5] Jagmohan Chauhan,[1,5] Yining Hu,[1,5] Kanchana Thilakarathna,[2,5] Diego Perino,[3] Dwight Makaroff,[4] and Aruna Seneviratne[1,5]
[1]University of New South Wales, [2]University of Sydney, [3]Telefonica Research, [4]University of Saskatchewan, [5]Data61/CSIRO-Australia

***Abstract*—The majority of available wearable computing devices require communication with Internet servers for data analysis and storage, and rely on a paired smartphone to enable secure communication. However, many wearables are equipped with WiFi network interfaces, enabling direct communication with the Internet. Secure communication protocols could then run on these wearables themselves, yet it is not clear if they can be efficiently supported.**

In this paper, we show that wearables are ready for direct and secure Internet communication by means of experiments with both controlled local web servers and Internet servers. We observe that the overall energy consumption and communication delay can be reduced with direct Internet connection via WiFi from wearables compared to using smartphones as relays via Bluetooth. We also show that the additional HTTPS cost caused by TLS handshake and encryption is closely related to the number of parallel connections, and has the same relative impact on wearables and smartphones.

***Index Terms*—Security, Wearables, Network Measurements**

I. Introduction

Wearables such as smartwatches, smartglasses and fitness bands, are becoming increasingly popular and were predicted in 2016 to become commodity devices in the same manner as smartphones [5], [11], [17]. These devices are equipped with a rich set of sensors that can continuously monitor a wide variety of attributes of the human body, physical surroundings and online user behaviours not available through any other means [16].

Many wearable applications (apps) upload users' personal information collected from body-worn sensors to cloud servers for analysis. Due to the sensitivity of the personal data collected by wearables, they must be transferred using a secure communication protocol. Today, most of the apps rely on a smartphone counterpart for communication with cloud services and in this way take advantage of secure communication protocols such as HTTPS. A wearable typically communicates with a paired smartphone via Bluetooth. Such communication is usually considered secure solely due to the low transmission range. However, the majority of wearables are already equipped with WiFi capability while Cellular networking is becoming a reality. These technologies enable direct Internet connectivity from wearables; it is thus vital to understand whether or not secure communication protocols can be efficiently realized by wearables directly.

In this paper, we address this critical question through an experimental study. As nearly 50% of today's Internet traffic is HTTPS, it is a natural candidate for secure communication for standalone wearables. Naylor *et al.* [13] show that the cost of HTTPS is not negligible even in the case of smartphones. We thus study the impact of HTTPS on wearables compared to HTTP in terms of the amount of *downloaded data*, *data transfer time* and *energy consumption*. We leverage two popular categories of wearables (smartglass and smartwatch), and take measurements from both a controlled web server and landing webpages of popular Internet websites. This allows us to characterize the resource consumption for each phase of the protocols precisely to derive the main factors contributing to the HTTPS performance difference, and to measure the impact when multiple parallel TCP connections and multiple external servers are used to access web resources from these devices.

We make the following main observations. First, the relative cost of HTTPS to HTTP in terms of *downloaded data* and *data transfer time* on wearables is comparable to that of smartphones, validating the fact that wearables are ready for HTTPS in terms of computing and networking capabilities. Second, because of the smaller battery capacity on wearables, the normalized *energy consumption* by the 100% battery capacity is higher on wearables than the smartphone. However, the overall *energy consumption* considering both wearables and smartphone is reduced (by $\sim$80% in the smartwatch and 100% in the smartphone) with direct and secure Internet communication from wearables primarily due to the elimination of the overhead of data exchange between the devices. Moreover, relaying incurs additional delay in real-time communication with external servers compared to the direct connectivity from wearables. Finally, we verify that the additional cost of HTTPS is mostly due to the TLS handshake (KEY exchange) phase and that the magnitude of the cost is closely related to the amount of data exchanged, and the number of parallel connections. These observations lead us to conclude that *wearables of today are ready for direct and secure internet communication via WiFi.*

The remainder of the paper is organized as follows. We first provide the background and discuss the related work. The experimental methodology is described next. We then present our analysis of HTTPS overhead in a controlled testbed environment followed by the results of web-browsing

experiments. Finally, we synthesize the implications of these findings into a set of recommendations that is followed by the conclusion and suggestions for future work.

II. Background and Related Work

A. Transport Layer Security (TLS) Background

The most widely used protocol to achieve secure communications over the Internet is HTTPS (HTTP over TLS) [10], [14]. We first describe the phases in HTTPS and HTTP connections followed by the TLS session negotiation procedure and the associated cryptographic algorithms.

Consider establishing a HTTPS connection (see Figure 1). The first phase is the *TCP handshake*, which is a three-way handshake between the server and the client. Next, server authentication and encryption key exchange takes place during the *TLS handshake*. Then the encrypted messages are exchanged during the *Data exchange* phase. Once message passing is finished, the TLS and TCP connections terminate in the *Connection termination* phase. An HTTP connection has three phases: *TCP handshake*, followed by *Data exchange* (plaintext messages), and *Connection termination*.

Next, we describe the *TLS handshake* phase in detail.

1) A TLS session is initiated by sending a *Client Hello* message to the server. The *Client Hello* message specifies the client's TLS capabilities, including the TLS version, cipher suites, a random number that will be used later to compute the final symmetric key, and compression method options. A cipher suite encompasses the choices/options for the cryptographic algorithms available for the different phases of the TLS session negotiation and data transfer: *key exchange algorithm*, *authentication algorithm*, *bulk cipher algorithm*, and *MAC (message authentication code) algorithm*.
2) Then, the server replies with a *Server Hello* message that contains the TLS version, cipher suite and compression method selected by the server from the mutually-supported choices offered by the client. Additionally, the server sends a random number in the message.
3) The server then sends messages containing *Certificate*, *Server Key Exchange* that includes server's public key, and a *Server Hello Done* message.
4) The client first verifies the server certificate. It then generates a pre-master secret that is encrypted using the server public key. The pre-master secret is then sent over to the server using a *Client Key Exchange* message.
5) The server decrypts the *Client Key Exchange* message using its private key to obtain the pre-master secret.
6) The server uses a combination of the pre-master secret, the random number it sent to the client in the *Server Hello* message and the random number it got from the client in the *Client Hello* message to compute the master secret. The client computes the master secret in the same manner.
7) The client sends a *Change Cipher Spec* notification to the server indicating that all subsequent messages will be authenticated and encrypted using the master secret. The client replies with an encrypted *Client Finished* message.
8) Finally, the server sends a *Change Cipher Spec* back to the client, completing the handshake. After the TLS negotiation phase is over, the encrypted application data is transferred between the client and the server.

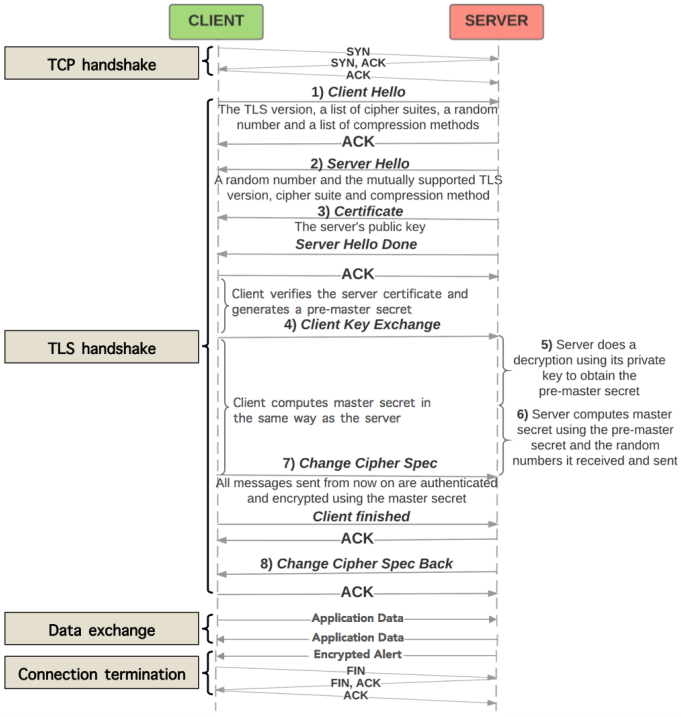


Fig. 1. HTTPS message sequence diagram with detailed TLS handshaking steps.

TLS Cryptographic Algorithms. Cryptographic Algorithms are comprised of four types of algorithms [2]:

Key Exchange Algorithm: The key exchange algorithm defines how the keys should be generated and exchanged between the client and server during the TLS session setup. The commonly-available Key Exchange algorithms include RSA (Rivest-Shamir-Adleman), DH (Diffie-Hellman), ECDH (Elliptic Curve Diffie-Hellman) and ECDHE (Elliptic Curve Diffie-Hellman Ephemeral). Both RSA and DH (as well as variations of DH) are asymmetric algorithms, meaning that they use two different keys: a public key for encryption and a private key for decryption. The advantage of Elliptic curve based algorithms over RSA is that they provide same security level as RSA with smaller key sizes. ECDHE provides PFS (perfect forward secrecy). PFS ensures that the sessions recorded in the past cannot be retrieved even if the server's long-term private keys are compromised.

Authentication Algorithm: The authentication algorithm determines if the client is communicating with the correct server entity. The authentication is generally done by exchanging certificates. RSA, Digital Signature Algorithm (DSA), and Elliptic Curve Digital Signature Algorithm (ECDSA) are the commonly used authentication algorithms. All the authentica-

tion algorithms are asymmetric and differ in the mathematical operations underlying the algorithm.

Bulk Cipher Algorithm: After the initial session setup phase, the user data between the client and the server is encrypted and decrypted using bulk cipher algorithms. All the bulk cipher algorithms are symmetric algorithms using the same key for encryption/decryption. The bulk cipher algorithms include block-based algorithms such as DES (Data Encryption Standard), 3DES, AES (Advanced Encryption Standard), and stream-based algorithms such as RC4 (Rivest's Cipher).

MAC Algorithm: The integrity of the application data is maintained using the MAC algorithm. It provides the means for a receiver to know that the application data is not altered en-route. The common MAC algorithms are MD5 (Message Digest 5) and SHA (Secure Hash Algorithm).

B. Related Work

Much previous research has investigated the performance of HTTPS on different computing platforms such as web servers [1], [4] and mobile devices [12], [13], [15], [18]. Apostolopoulos *et al.* [1] measured the performance of TLS using the SPECWeb96 benchmark on Apache and Netscape web servers. Their results showed that the use of TLS decreases the number of web transactions that a server can handle by a factor of two. A comprehensive study by Coarfa *et al.* [4] on TLS web servers revealed that RSA operations incur the largest performance cost. The study also found out that RSA accelerators are effective for e-commerce type workloads as they experience low TLS session reuse.

Youngsang *et al.* [18] compared the performance of the TLS protocol between PDAs and laptops in secured WEP and open WiFi environments. The studies confirmed that the additional delay for TLS handshake is three times more on a PDA than for a laptop. However, the major factor is not cryptographic computation but network latency and other PDA architecture issues. In other work, Naylor *et al.* [13] studied the collected dataset from large ISPs to examine the world-wide adoption rate of HTTPS. The study indicated that HTTPS adds significant latency in the mobile networks and that the cost of TLS on smartphones is not negligible.

Miranda *et al.* [12] measured the energy consumption of TLS on Nokia N95 mobile phones on WiFi and 3G networks and showed that the energy required for transmission and I/O during TLS handshake far exceeds the energy spent on computing cryptographic operations. Potlapally *et al.* [15] performed a comprehensive performance analysis by studying the impact of various TLS cryptographic algorithms with varying bit sizes on the energy consumption of a PDA device. The main findings of the study suggested that the asymmetric and hash algorithms have the highest and the least energy costs, respectively, and a better trade-off between the encryption and energy consumption can be achieved by tuning parameters such as the key size.

Our study differs from previous work as we are expanding the context of evaluating web security protocol overhead with respect to the wearable platform. To the best of our knowledge, we are the first to quantify the performance of HTTPS vs. HTTP experimentally in terms of *energy consumption*, *data transfer time* and *downloaded data* on weafrables and compare them to HTTPS performance on smartphones.

TABLE I
HARDWARE CHARACTERISTICS OF THE DEVICES.

Device	CPU	Memory	Battery
Watch	Quad-core processor 1.2 GHz	512 MB	410 mAh
Glass	Dual-core processor 1 GHz	2 GB	570 mAh
Phone	Quad-core processor 1.5 GHz	2 GB	2100 mAh

III. EXPERIMENTAL METHODOLOGY

We first present the devices used in the experiments, followed by the scenarios and the evaluation metrics.

A. Devices

We select two devices representative of two popular wearable categories: smartglasses and smartwatches. Specifically, we use Google Glass Explorer edition running Android XE 18.11 as an example of smartglasses, and LG G Watch R running Android Wear 5 as an example of smartwatches. Baseline results are obtained from a Nexus 4 smartphone running Android 5. All communication to the Internet is done via WiFi while the smartphone and smartwatch communicate via Bluetooth Classic. The hardware specifications for all the devices is shown in Table I.

B. Experimental Scenarios

We perform two sets of experiments; 1) within a controlled testbed of a customized web server and client-side application, and 2) with publicly available Internet web servers and existing client applications. We focus on file download only (i.e., GET), as the same observations and trends also hold true for the file upload (i.e., POST).

Controlled Testbed. We run a web server in the local network that supports HTTP and HTTPS file requests. The server runs Apache server 2.4.7 on Ubuntu 14.04; we disable caching and page encoding by setting the specific headings in the downloaded PHP files. On the client side, we develop a custom mobile/wearable app that is compatible with all the three devices, and can generate a single GET or single POST request to the web server. We were not able to leverage existing wearable apps as they do not allow the user to select between HTTP and HTTPS protocols, nor to collect performance statistics. We run the following two sets of experiments in the controlled testbed. Each experiment is repeated 30 times and we present cumulative results averaged over all the runs.

1) Exp1: downloading a single file (GET request) from the local web server, which allows us to characterize the cost associated with different phases of a HTTP/HTTPS session precisely. We also vary the file size to understand the main factors contributing to this cost on wearables. For this experiment, as the reference TLS parameters, we use TLS 1.0 version with the following set of algorithms: ECDHE for key

TABLE II
WEBSITES USED IN OUR EXPERIMENTS.

Website	Url	ID
Bing	www.bing.com	B
Apple	www.apple.com	A
VK	www.vk.com	V
Terraclicks	www.terraclicks.com	T
Java	www.java.com	J
Douban	www.douban.com	D

exchange, RSA for authentication, AES 128 for bulk cipher, and SHA for MAC.

2) Exp2: downloading a single file from the local web server with different combinations of TLS cryptographic algorithms. We vary the supported TLS algorithm combinations one by one and tested them on a fixed file size of 100 KB. As there are many options available for each of the four major types of algorithms involved in TLS, we focus on the ones frequently used on modern devices. For the key exchange algorithm, we select RSA (2048 bits) and ECDHE (65 bytes). The former is one of the oldest algorithm developed for key exchange and is still in wide use [8]. The latter is the preferred choice on modern devices and web servers. For the authentication algorithm, we select RSA, since RSA signed certificates are the most used on the web. We do not consider ECDSA as the usage of ECDSA signed certificates on the web is still in a very earlly stage. We choose to experiment with AES (128 and 256 bit) and RC4 for bulk cipher algorithm. We test RC4 to compare stream-based algorithms against block-based algorithms such as AES. For the MAC algorithm, we choose SHA. Modern web servers use AES as bulk cipher algorithm and SHA as the MAC algorithm, as they are the most secure algorithms available in their respective categories [7], [19].

Internet Experiments. We investigate the impact of HTTPS on wearables in realistic settings with public web servers and existing web browsers. Specifically, Exp3 allows us to measure the impact of TLS when multiple parallel TCP connections and multiple external servers are used to access web resources from the wearables. Each experiment is repeated 5 times and we present the average results of all the runs.

3) Exp3: We select a subset of websites from the Alexa top 500[1] and download the landing webpages of those websites. The selected websites support both HTTP and HTTPS and transfer all objects using one of the two protocols. It turns out that only 6 websites satisfy our requirements (Table II), while the others either support only one of the two protocols, or use a mix of HTTP and HTTPS for the data exchange. As our goal is to understand the cost of HTTPS with multiple connections and servers and not to provide an in-depth analysis of the impact of HTTPS on user web experience, we believe these websites are sufficient for Exp3. On the client side, we use web browsers as apps because we need to parse the website's entire root page, and recursively fetch embedded objects. We also disable HTTP/2 in order to avoid long-lived HTTPS connections [9], and caching. Specifically, we use Android WebView[2] for the smartphone and smartglasses, and Android Wear Internet Browser[3] for the smartwatch as WebView is not supported on the smartwatch.

[1]http://www.alexa.com/topsites

C. Evaluation Metrics

We consider the amount of *downloaded data*, the *data transfer time* and the *energy consumption* as the main metrics for our analysis. To measure the *downloaded data* and *data transfer time*, we first run `tcpdump` on each device to capture the packets for each experiment. We then extract the total number of bytes downloaded, as well as the time taken by each phase of the HTTP/HTTPS connections by analyzing the TCP packet transmissions between the server and the client.

We consider both the raw energy usage and the energy consumption normalized by the battery capacity of each device. Raw energy consumption is measured for every experiment with a Monsoon power monitor[4] directly connected to each device via USB. Energy is obtained by integrating the instantaneous power values calculated using current and voltage measurements sampled at 0.2 ms time intervals. We carry out all the experiments with a fully charged battery to prevent additional current being drawn from the power monitor. Hence, the current drawn from the power monitor is only required to perform the task associated with the experiments and background processes. The exact energy usage of the experiment is computed by deducting the energy utilization of the background processes. Background process energy use is approximately constant when the device is in the idle state.

IV. CONTROLLED TESTBED

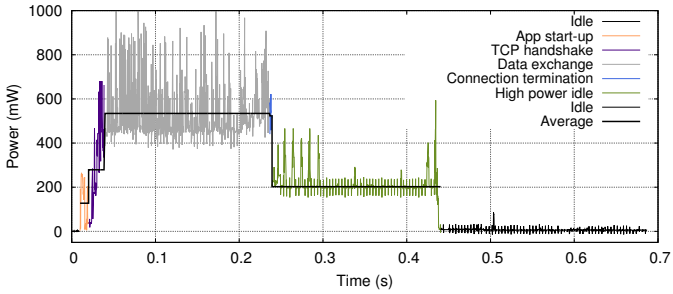
In this section, we present a set of experiments performed in a controlled testbed to characterize the impact of TLS on wearables. We first present a detailed analysis of a single file exchange for different sizes (Exp1), and then investigate the influence of TLS algorithms (Exp2).

A. Understanding the Cost of TLS

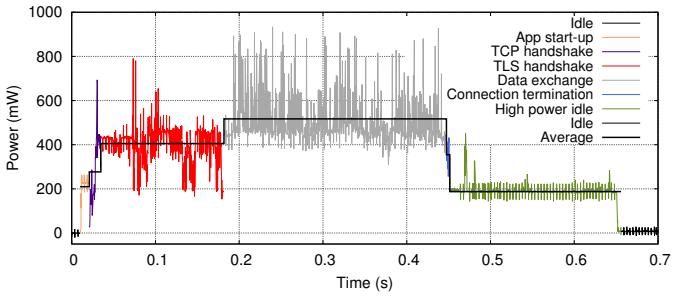
As an illustrative example, Figure 2 shows an instance of the power consumption profile for the smartwatch when downloading a 500 KB file from the local web server. The profile shows the different phases of the data transfer, namely *TCP handshake*, *TLS handshake*, *Data exchange*, and *Connection termination*. Additionally, we can observe *Application start-up* and *High power idle* phases in the power consumption profile.

We clearly observe that each phase has a different characteristic value for each metric. The highest power level is reached during the *Data exchange* phase for both HTTP and HTTPS, on average 534.12 mW and 516.38 mW respectively. This cost is primarily due to the active radio transmitters/receivers. The power profile for the data exchange phase is closely correlated with the data transfer rate. By observing the elapsed time

[2]http://developer.android.com/reference/android/webkit/WebView.html
[3]https://play.google.com/store/apps/details?id=com.appfour.wearbrowser
[4]https://www.msoon.com/LabEquipment/PowerMonitor/



(a) LG G Watch R power profile for HTTP.



(b) LG G Watch R power profile for HTTPS.

Fig. 2. Energy consumption profile when downloading a 500 KB file.

and bytes downloaded, we can infer a lower downlink data rate for HTTPS than HTTP due to the processing overhead of encrypting data (i.e., 2.1 Mbps vs. 2.6 Mbps). Despite the lower average power, the total power consumption for data exchange is larger for HTTPS, because of the encryption and the longer data exchange time due to the combination of additional encryption bits and lower data rate.

TCP connection management (i.e, *TCP handshake* and *Connection termination*) consumes very little energy as it is composed of only a few packets. Energy consumption for *Application start-up* is not negligible, although there is no data exchanged during this period. *High power idle* profile ($\sim$200 mW for $\sim$200 ms) is similar for both HTTP and HTTPS as it is independent of the higher layer protocols.

As explained in the background section, *TLS handshake* is an additional phase in HTTPS where the client and server agree upon the TLS attributes for the session and exchange keys for payload encryption/decryption. Although the TLS handshake generates limited data, it has a significant energy footprint as this phase takes about 147 ms and 405.22 mW on average; we further examine the performance details in the following paragraphs.

Figures 3(a) and 3(b) present the average values of *data transfer time* and *energy consumption* for each phase of the HTTP/HTTPS connection respectively, while Table III reports the *downloaded data* in bytes for each phase. Overall, as expected, we observe that TLS increases the *data transfer time* by $\sim$75% to $\sim$90%, the amount of *downloaded data* by $\sim$2% and the *energy consumption* by $\sim$70% to $\sim$100% for all the devices when downloading a 500KB file.

Also as expected, the large file size used in this experiment

TABLE III
DOWNLOADED DATA AMOUNT (BYTES) FOR DIFFERENT CATEGORIES.

Protocol	TCP hand.	TLS hand.	Data down.	Connection term.
HTTP	74	-	523212	66
HTTPS	74	5421	524390	132

TABLE IV
$\frac{\text{HTTPS}}{\text{HTTP}}$ RATIO OF DOWNLOAD DATA AMOUNT (BYTES) FOR DIFFERENT TLS PARAMETERS.

RSA+ RC4	RSA+ AES128	RSA+ AES256	ECDH+ RC4	ECDH+ AES128	ECDH+ AES256
1.0515	1.0524	1.0524	1.0548	1.0557	1.0557

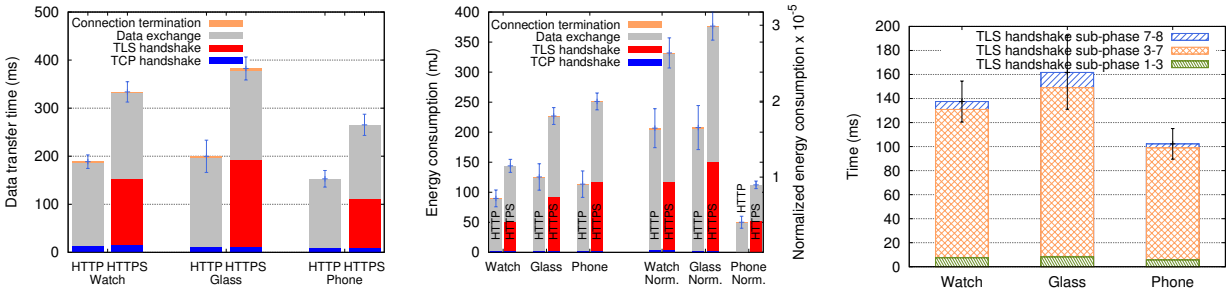
makes the impact on the total amount of data exchanged of the other phases negligible as can be seen from Table III. The smartphone provides the best performance in terms of *data transfer time* followed by smartwatch and smartglass (cf. Figure 3(a)). This reflects the hardware capabilities of the different devices, which in turn influences the *energy consumption* as shown in Figure 3(b). On the one hand, larger devices consume more power as we observe on the left-hand side (from the middle) of Figure 3(b). On the other hand, as the battery capacity of smartphone is larger than their wearable counterparts, the relative energy consumption is more significant on the wearables as shown on the right-hand side (from the middle) of Figure 3(b). To further validate our results, in particular our baseline, we compared the energy consumption and data transfer time of smartphone with the previously reported results. Miranda *et al.* [12] showed that the energy values are between 100 mJ and 150 mJ per HTTPS transaction using the same TLS parameters as in our experiments; whereas our average energy consumption value is 128 mJ. Similarly, the data transfer times in [12] are between 100 ms and 200 ms whilst our results show 110 ms average.

We further analyze the TLS handshake process in detail for all the devices in Figure 3(c). We make two main observations. First, the different duration of the TLS handshake across devices reflects their hardware characteristics. Second, certificate validation and key computation on the client side (*TLS handshake* sub-phases 3 through 7, as explained in the background section) is taking $\sim$87% of the total time of *TLS handshake*.

Take-away message: *The energy consumption and data transfer time during the costly TLS handshake phase depend on the hardware capabilities of the device. As expected, the cost of TLS is more significant in the lower resourceful wearable devices than in the smartphones.*

B. Impact of the Wireless Communication Technology

Bluetooth communication is utilized if smartwatch apps leverage the smartphone to connect to the Internet (BT relaying), as this is the chosen method for most current applications [3]. We measured the *energy consumption* and *data transfer time* for BT relaying via Bluetooth, and compare them to direct



(a) Data transfer time. (b) Energy consumption. (c) TLS phases duration.

Fig. 3. Cost of HTTP and HTTPS for a 500 KB file download (i.e., GET).

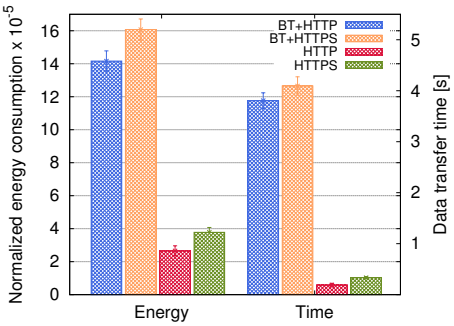


Fig. 4. Wireless technology cost.

Internet communication using HTTP and HTTPS over WiFi. We assume that the smartwatch and smartphone are already paired and are less than 1 metre apart for BT relaying. Figure 4 shows the smartwatch *energy consumption* normalized by 100% battery capacity and the *data transfer time*. The *energy consumption* of this additional data exchange between smartwatch and smartphone is ∼6 times higher than via WiFi on a smartwatch. Also, Bluetooth relaying adds ∼4 seconds of additional *data transfer time*. WiFi guarantees the minimal power/throughput ratio [6], and is thus more power efficient than Bluetooth for file transfer. Also, the Bluetooth transfer significantly increases the transmission delay because of the lower transmission rate compared to WiFi and the additional relaying with the smartphone.

Take-away message: *Direct Internet communication reduces energy consumption and data transfer time compared to the current norm of Bluetooth relaying.*

C. Impact of Transfer File Size

The effect of TLS protocol selection on the varying sizes of downloaded files is shown in Figure 5. Overall, the $\frac{\text{HTTPS}}{\text{HTTP}}$ ratio is comparable across devices and decreases as the file size increases for all the metrics: the larger the file, the more the cost of TLS handshake is amortized.

The ratio of *data transfer time* ($\frac{\text{HTTPS}}{\text{HTTP}}$) is highest in thesmartglasses followed by the smartwatch and smartphone for all file sizes (Figure 5(a)). However, the magnitude of the *download bytes* ratio ($\frac{\text{HTTPS}}{\text{HTTP}}$) is almost identical for all three devices as they use the same TLS parameters (Figure 5(b)). Finally, Figurse 5(c) and 5(d) show the impact of TLS to be higher on the phone, but the impact is relatively lower as the phone battery has a larger total capacity. Also, despite the higher $\frac{\text{HTTPS}}{\text{HTTP}}$ *energy consumption* ratio for small files, the relative impact of these protocols is higher for large files because of the longer data transfer duration and encryption/decryption being done for more bytes.

Take-away message: *Despite the lower resources in wearables, the relative cost of enabling secure direct communication on wearables is comparable to smartphones.*

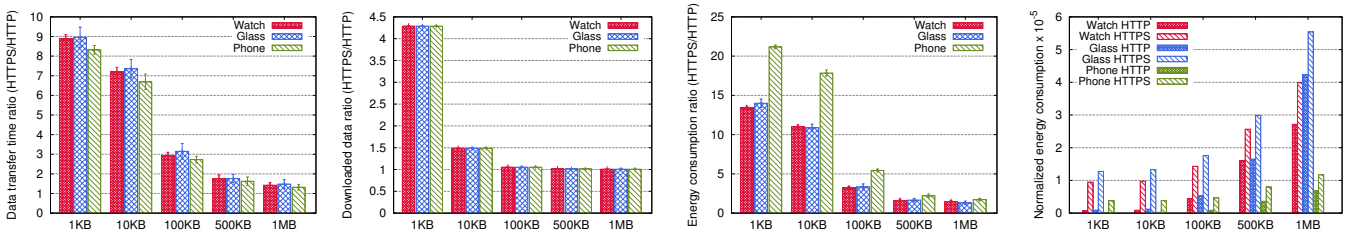
D. Impact of Cryptographic Algorithms

The results of *data transfer time* ratio and *energy consumption* ratio obtained by running different combinations of cryptographic algorithms are shown in Figure 6(a) and 6(b), while Table IV shows the *downloaded bytes* ratio. Moreover, Figures 6(c) and 6(d) show the effect of changing TLS parameters on *Data exchange* and *TLS handshake* phases. As expected, various cryptographic algorithms have different impacts on HTTPS performance for all the devices. Overall, changing cryptographic parameters is less significant than modfying the other parameters (e.g., file size).

Using ECDHE instead of RSA leads to an increase in the *downloaded bytes*, *data transfer time*, and the *energy consumption*. As mentioned earlier, the key exchange algorithm has a significant impact on the key exchange phase for all the three devices. For example, changing the key exchange algorithm from (RSA+RC4) to (ECDHE+RC4) leads to a ∼7% increase in *downloaded bytes*, ∼10% increase in *data transfer time* and ∼1% increase in *energy consumption* on the smartwatch's key exchange phase. This is because ECDHE uses more complex mathematical operations than RSA to perform the encryption/decryption.

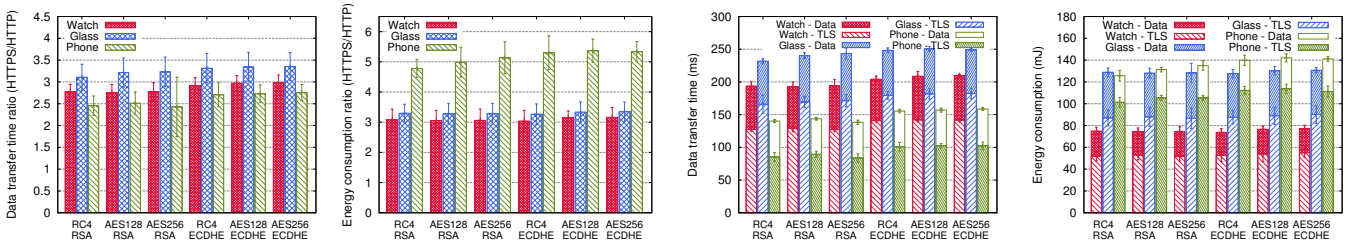
The choice of bulk cipher algorithm in TLS does not have a significant impact on the metrics. For example, changing the bulk cipher algorithm from (RSA + RC4) to (RSA + AES128) leads to ∼0.1% increase in *downloaded bytes*, ∼1% increase in *data transfer time* and ∼1% increase in *energy consumption* on the smartwatch's application data transfer phase.

Take-away message: *Changes in the cryptographic parameters do not have a significant impact on the considered*



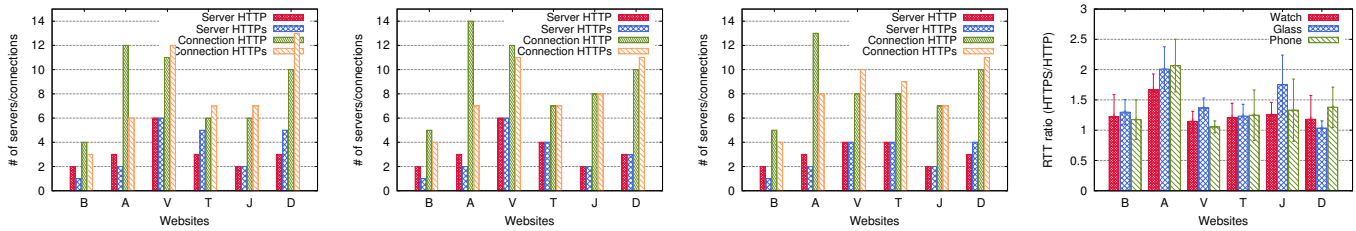
(a) Data transfer time. (b) Data usage. (c) Energy consumption. (d) Normalized energy.

Fig. 5. Impact of file size on data transfer time, data downloaded and energy consumption.



(a) Data transfer time. (b) Energy consumption. (c) Data transfer time. (d) Energy consumption.

Fig. 6. Impact of TLS parameters on data transfer time and energy consumption.



(a) Watch (b) Glass (c) Phone (d) Device RTT

Fig. 7. (a)-(b)-(c) Analysis of the number of connections/servers involved in data exchange between mobile devices and Internet websites. Refer to Table II for website identifiers. (d) Ratio of RTT values $\frac{\text{HTTPS}}{\text{HTTP}}$.

evaluation metrics.

V. Internet Experiments

In this section, we present a set of experiments performed with Internet websites to understand the impact of TLS when multiple parallel TCP connections and multiple external servers are involved in a data exchange. We first analyze the properties of data exchange when mobile devices interact with the web content (i.e., number of connections and servers involved in the data exchange and RTT). We then investigate the role of the main data exchange parameters (i.e. number of servers, number of connections) on the TLS cost. For all experiments in this section, we follow the experimental methodology Exp3, using the websites reported in Table II.

A. Understanding Data Exchange with Web Servers

We start our analysis by investigating the main characteristics of the data exchange between the devices and external websites. The goal is to identify the main parameters that may affect the cost of HTTPS and quantify their effect.

Figures 7(a), 7(b), and 7(c) depict the average number of connections and servers involved in each data exchange for HTTP and HTTPS for each device type. As expected, we observe the number of servers and connections involved in the communication varies between websites. Specifically, the numbers of servers ranges between 1 and 7, while the number of connections is between 4 and 13. Indeed, those parameters are specific to each website and are related to the number of embedded objects and the servers from which they are delivered to the users.

We also notice that the same websites on different devices use a different number of connections and servers, even for the same protocol. For instance, `https://www.apple.com` (A) opens 6, 7, and 8 connections for HTTPS in smartwatch, smartglass and smartphone respectively. This is due to two main factors. First, websites provide different versions to users according to the device type. These versions may differ on the data exchange parameters (e.g., servers, connections), but can also differ in the actual content (e.g., embedded objects, main

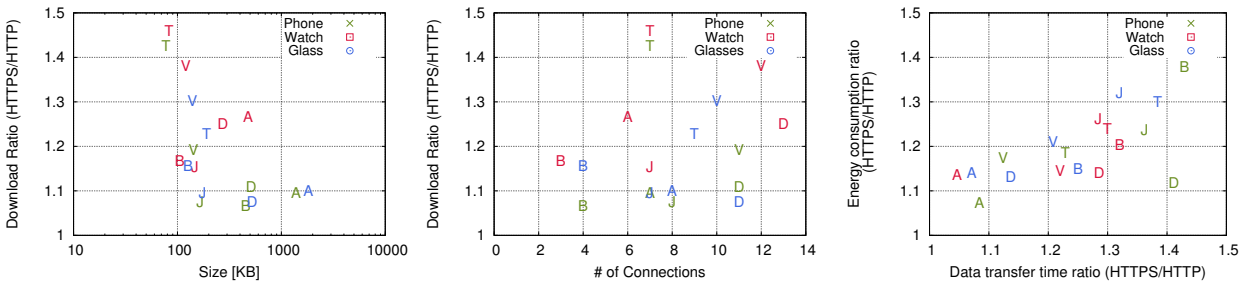


Fig. 8. TLS cost distribution of websites in the three devices. The marker identifies a website (Table. II) and the device.

page). Second, different devices and browsers have a different level of support for websites' features (e.g., Javascript, Flash).

B. Analysis of the Cost of TLS in the Wild

Figures 7(a), 7(b), and 7(c) show that the number of servers and connections varies between HTTP and HTTPS for most websites due to the reasons that were explained in the previous subsection. It is not possible to identify a clear trend, however, as the number of servers and connections increases with TLS for some websites while it decreases for others. As an explanation for this behaviour, we note that the number of connections basically depends on TCP connection reusability, where if there is an existing open connection, the browser tries to use that connection to fetch a new web object rather than opening a new connection. Most websites have different data exchange parameters for HTTP and HTTPS, and that those parameters are specific to each website.

We report the average Round Trip Time (RTT) ratio ($\frac{\text{HTTPS}}{\text{HTTP}}$) for each website in Figure 7(d). The RTT value always increases with HTTPS (i.e., from ∼10% to ∼90%), due to the additional processing overhead at both the servers and the clients required for data encryption/decryption. Intuitively, we expect low-resource wearables to have a larger increase in RTT, which is not what we observe, as the increase is approximately similar for all devices. This suggests that HTTP/SSL computation on low-resource client devices has a lower impact on the RTT than on the server side.

Figure 8(a) depicts the ratio ($\frac{\text{HTTPS}}{\text{HTTP}}$) of *downloaded bytes* for each website and device. The larger the size of the website, the lower the cost of TLS in terms of *downloaded bytes* (i.e., the ratio is ∼1.5 when the file size is 100 KB and ratio goes down to ∼1.05 when the file size increases up to 1 MB). This confirms that the trend noticed in Exp1 also holds true for the data exchange with Internet websites. Among the measured websites, `www.terraclicks.com` (T) has the smallest size, yet TLS has the greatest performance impact. Moreover, as highlighted in the previous section, since websites provide different content for different devices and hence use a different number of connections in those devices, the volume of data exchanged is different. Therefore the *downloaded bytes* ratio ($\frac{\text{HTTPS}}{\text{HTTP}}$) varies across devices for the same website.

In Figure 8(b), we show the relation between the number of connections and the *downloaded bytes* ratio ($\frac{\text{HTTPS}}{\text{HTTP}}$) for each website across devices. As expected, due to extra bytes added by the TLS handshake for each and every connection in HTTPS, the $\frac{\text{HTTPS}}{\text{HTTP}}$ ratio increases with the number of connections (i.e., ∼1.05 for 4 connections to ∼1.4 for 12 connections). However, in cases such as `www.terraclicks.com` (T), the byte overhead is dominated by the small size of the website rather than the number of connections. Due to the different sizes of certificates and different configurations of TLS parameters supported by different servers, the number of extra bytes added to each connection varies. This causes slight deviations to the expected trend for the ratio ($\frac{\text{HTTPS}}{\text{HTTP}}$) vs. number of connections.

As observed in Exp1, the *data exchange time* ratio impacts the *energy consumption* ratio the most. Therefore, we analyze the *energy consumption* ratio (Figure 8(c)) as a linear function of the *data exchange time* ratio (in the range ∼1.05 to ∼1.5). As highlighted in Exp1, TLS handshake time and lower data transferring rate in HTTPS are the major causes of the longer duration in HTTPS.

Take-away message: *The impact of the device type for data transfer time is less when accessing the Internet servers than the controlled testbed, due to the fact that the server side latency is more significant than in the client side. Moreover, the impact of TLS on the evaluation metrics have the similar behavior in both controlled testbed and Internet results.*

VI. Main Findings and Recommendations

The main finding of the paper is that *secure direct Internet connectivity via WiFi on wearables is feasible, improves performance and reduces energy footprint.* In light of that, we provide the following recommendations to wearable application/system developers:

- Although wearables have the ability to connect to the Internet directly, most current wearable applications leverage the smartphone to connect to Internet. However, this requires the transfer of data from wearables to the smartphone via Bluetooth, and results in an increase in both *data transfer time* and *energy consumption*.
  By using direct secure Internet communication instead of Bluetooth relaying for a typical website download, it is possible to achieve ∼78% energy savings on the wearable (as well as 100% in the smartphone) and reduce elapsed time by ∼40%. *Therefore, we recommend that app developers should utilize direct Internet connectivity*

via WiFi from wearables eliminating the additional costly step of Bluetooth relaying.

- Despite the resource constraints on wearables, the relative cost of HTTPS, (i.e. the $\frac{\text{HTTPS}}{\text{HTTP}}$, ratio) in terms of *data transfer time* is comparable on wearables and smartphones for all the sizes. The amount of *downloaded data* is the same for any size of data download, if all the TLS algorithms used are the same for all the devices. Thus, standalone secure wearable applications can be practically realized utilizing the existing secure protocols such as HTTPS.

Moreover, we have the following observations that hold true for both smartphones and wearables.

- As expected, the impact of HTTPS is mostly driven by the size of the object (e.g., webpage, file) and the number of TCP connections, and it is dominated by the TLS handshake processing and delays. For small objects, TLS induces a significant increase in the *energy consumption*, the *data transfer time*, and the volume of *downloaded data*. However, for large objects, the TLS cost is relatively smaller, and can be in the order of 15%. These costs also increase with the number of connections and the servers. The TLS handshake is the main contributor to the additional cost of HTTPS, while data encryption/decryption has a much lower impact. Therefore, the traditional approaches used in the mobile communications such as bulk transferring and reducing the number of TLS sessions can be exploited to improve the performance.
- The choice of TLS algorithms has a limited impact on the overall TLS cost. Key exchange is the stage that can be influenced the most by the choice of the algorithm and therein lies a security/performance trade-off. Conversely, different bulk cipher algorithms provide similar performance, and the one guaranteeing stronger security can thus be selected without adding much overhead.

VII. Conclusion

In this paper, we answered the critical question – "Are wearables ready for HTTPS?" – through a systematic experimental measurement study using of-the-self wearable devices. We ran our experiments of downloading files over HTTP and HTTPS from an internal server that we have control over and also downloading 6 of the most popular websites according to Alexa ranking that support both HTTP and HTTPS. We consider the amount of *downloaded data*, the *data transfer time*, and the *energy consumption* as the main metrics for our analysis.

The experimental results showed that *current wearables are ready* for HTTPS, i.e. for secure and direct Internet connectivity. In fact, direct Internet communication via WiFi reduces energy consumption and improves performance compared to the current norm of Bluetooth relaying through a smartphone. Therefore, we advocate that wearable app/system developers should leverage these advanced capabilities of wearables to drive a paradigm shift toward standalone wearable devices. We also showed that relative impact of securing the connection in wearable is similar to smartphones and provided recommendations to app/system developers to further optimize efficiency and improves security using existing techniques.

Finally, important directions for future work are to study the cost of HTTPS in the cellular interface when it becomes common in consumer wearables, and to evaluate the performance of other commonly used protocols such as HTTP/2 and QUIC on wearables.

References

[1] G. Apostolopoulos, V. Peris, and D. Saha, "Transport Layer Security: How Much Does It Really Cost?" in *INFOCOM*, Mar 1999, pp. 717–725.

[2] D. Berbecaru, "On measuring ssl-based secure data transfer with handheld devices," in *2005 2nd International Symposium on Wireless Communication Systems*, Sept 2005, pp. 409–413.

[3] J. Chauhan, S. Seneviratne, M. A. Kaafar, A. Mahanti, and A. Seneviratne, "Characterization of early smartwatch apps," *2016 IEEE International Conference on Pervasive Computing and Communication Workshops, PerCom Workshops 2016*, no. May, 2016.

[4] C. Coarfa, P. Druschel, and D. S. Wallach, "Performance Analysis of TLS Web Servers," *ACM Trans. Comput. Syst.*, vol. 24, no. 1, pp. 39–69, Feb. 2006.

[5] T. Danova, "THE WEARABLES REPORT: Growth trends, consumer attitudes, and why smartwatches will dominate," pp. 1–4, 2014. [Online]. Available: http://www.businessinsider.com/the-wearable-computing-market-report-2014-10?op=1

[6] R. Friedman, A. Kogan, and Y. Krivolapov, "On Power and Throughput Tradeoffs of WiFi and Bluetooth in Smartphones," *IEEE Trans. on Mob. Comput.*, vol. 12, no. 7, pp. 1363–1376, Jul. 2013.

[7] P. Hoffman and B. Schneier, "Attacks on cryptographic hashes in internet protocols," Tech. Rep., 2005.

[8] L.-S. Huang, S. Adhikarla, D. Boneh, and C. Jackson, "An experimental study of tls forward secrecy deployments," *IEEE Internet Computing*, vol. 18, no. 6, pp. 43–51, 2014.

[9] IEFT, "RFC 7540 - Hypertext Transfer Protocol Version 2 (HTTP/2) http://tools.ietf.org/html/rfc7540," pp. 1–96, 2015. [Online]. Available: https://tools.ietf.org/html/rfc7540

[10] IETF, "The Transport Layer Security (TLS) Protocol Version 1.2," https://tools.ietf.org/html/rfc5246.

[11] X. Liu, T. Chen, F. Qian, Z. Guo, F. X. Lin, X. Wang, and K. Chen, "Characterizing smartwatch usage in the wild," in *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '17. ACM, 2017, pp. 385–398. [Online]. Available: http://doi.acm.org/10.1145/3081333.3081351

[12] P. Miranda, M. Siekkinen, and H. Waris, "TLS and Energy Consumption on a Mobile Device: A Measurement Study," in *ISCC*, Jun. 2011, pp. 983–989.

[13] D. Naylor, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Marco, M. Munafò, K. Papagiannaki, and P. Steenkiste, "The cost of the 'S' in HTTPS," in *CoNeXT*, Dec. 2014, pp. 133–140.

[14] S. Petridou and S. Basagiannis, "Towards energy consumption evaluation of the ssl handshake protocol in mobile communications," in *2012 9th Annual Conference on Wireless On-Demand Network Systems and Services (WONS)*, Jan 2012, pp. 135–138.

[15] N. Potlapally, S. Ravi, A. Raghunathan, and N. Jha, "A study of the energy consumption characteristics of cryptographic algorithms and security protocols," *IEEE Trans. on Mob. Comput.*, vol. 5, no. 2, pp. 128–143, Feb 2006.

[16] E. Sazonov and M. R. Neuman, *Wearable Sensors: Fundamentals, implementation and applications*. Elsevier, 2014.

[17] S. Seneviratne, Y. Hu, T. Nguyen, G. Lan, S. Khalifa, K. Thilakarathna, M. Hassan, and A. Seneviratne, "A survey of wearable devices and challenges," *IEEE Communications Surveys Tutorials*, vol. PP, no. 99, pp. 1–51, 2017.

[18] Y. Shin, M. Gupta, and S. Myers, "A Study of the Performance of SSL on PDAs," in *INFOCOM Workshops*, Apr. 2009, pp. 1–6.

[19] G. Singh, "A study of encryption algorithms (rsa, des, 3des and aes) for information security," *International Journal of Computer Applications*, vol. 67, no. 19, 2013.