Towards Utilizing Tcpcrypt in Mobile Healthcare Applications

Stefanos A. Nikolidakis, Vasileios Giotsas, Emmanouil Georgakakis, Dimitrios D. Vergados, and Christos Douligeris

Department of Informatics. University of Piraeus
80, Karaoli & Dimitriou St., GR-185 34, Piraeus, Greece
{snikol,egeo,vergados,cdoulig}@unipi.gr,
giotsas@ieee.org

Abstract. The evolution and growth of networks has made the personal data of the users available to many applications. In this direction, one of the main concerns is to protect the sensitive personal information, while at the same time avoid delays in the provision of services like healthcare to the general public. An extension of TCP, the Tcpcrypt, is a promising technology that can be used on this field. Tcpcrypt is designed to provide end-to-end encryption in the transport layer with low overhead, rendering it a very promising solution in order to protect medical data that are often handled by devices with limited resources. In this paper Tcpcrypt performance is evaluated against TCP, in terms of additional overhead incurred in the total size of the transmitted data and the total number of CPU instructions that are executed. Moreover, a solution for reducing overhead through fine-grained packet handling is proposed.

Keywords: Tcpcrypt, SSL, Healthcare.

1 Introduction

Preserving the privacy and the integrity of data transmitted over networks is a well established requirement and many solutions have been proposed and implemented towards this direction. Security and encryption mechanisms can be deployed in the upper layers of the network stack. Some of the most widely used solutions to provide authentication and encryption mechanisms are SSH (Secure Shell) and Https [1, 2] which are deployed in the application layer, TLS (Transport Layer Security) / SSL (Secure Sockets Layer) [3] are deployed in the transport layer and IPsec [4], is deployed in the Internet Layer. Tcpcrypt has emerged as an alternate solution in the transport layer that will address some of the shortcomings of the existing technologies [5].

Tcpcrypt enhances TCP by adding cryptographic capabilities. One of the key benefits of Tcpcrypt is transparency as it requires no configuration, no changes to applications and the network connections will continue to work even if the remote end does not support Tcpcrypt. In the latter case the connections will gracefully fall back

K.S. Nikita et al. (Eds.): MobiHealth 2011, LNICST 83, pp. 374–379, 2012.

to standard clear-text TCP. Tcpcrypt operates in the transport layer. In [6] a comparison of the performance of Tcpcrypt, TLS and SSL in terms of the number of connections a server can handle per second and the possible transfer rate was provided. In both metrics Tcpcrypt appears to have superior performance. One of the reasons Tcpcrypt is less demanding comparing to TLS/SSL is the fact that it does not utilize asymmetric cryptography mechanisms, which are computationally demanding operations. Moreover, the need of digital certificates and some form of PKI (Public Key Infrastructure) and CA (Certificate Authority) is essential for the use of SSL rendering its deployment cumbersome. The use of digital certificates enables SSL to perform strong authentication of the involved entities. Tcpcrypt authentication mechanisms cannot defend against active attacks. However Tcpcrypt can rely on application level authentication to ensure proper authentication and does not specify the means of the authentication e.g. certificates, passwords, tokens etc. Tcpcrypt is vulnerable to active attacks such as Man In the Middle Attacks (MIMA). For example, an attacker can modify a server's response to claim that Tcpcrypt is not supported (when in fact it is) so that all subsequent traffic will be transmitted in clear text and be susceptible to eavesdropping.

Given the promising capabilities of Tcpcrypt it is worth-investigating the performance of Tcpcrypt in mobile healthcare applications. A widely adopted paradigm entails a WBAN (Wireless Body Area Network) which collects and transmits data to a mobile sink attached to the patient. The sink is usually a IEEE802.11 capable mobile device which handles the communication with a healthcare server or other sinks in an ad-hoc manner. The data transmitted by the sink should be protected from malicious attackers, but at the same time the sink has to maintain low power consumption to achieve the longest possible availability. Tcpcrypt can become a severe handicap for the expected battery lifetime. Until now there is no published attempt to characterize the overhead incurred by Tcpcrypt at the client side. In this paper, such an attempt is presented and the realistic conditions under which Tcpcrypt can be deployed on mobile resource-limited devices are provided.

The paper is organized as follows: In Section 2.1, the comparison of transmitted bytes with the use of TCP and Tcpcrypt as the file size increases is presented. In Section 2.2, the CPU utilization and total duration for the transmission of data is depicted. In Section 2.3, the results of the previous sections are discussed. In Section 2.4, the reducing overhead through fine-grained packet handling is suggested. Finally, in 3, conclusions are given and future work is discussed.

2 Tcpcrypt Overhead Evaluation

The performance of Tcpcrypt against TCP, in terms of additional overhead incurred in the total size of the transmitted data and the total number of CPU instructions executed, are evaluated and compared in this section. Currently there is no Tcpcrypt implementation for ARM architecture (Advanced RISC Machine), thus it is not possible to evaluate it on handheld devices (e.g. android or iphone smart phones).

Therefore, the user-level implementation of Tcpcrypt protocol on a single-core Intel U3500 CPU (1.4 GHz) netbook with 2GB of RAM that operated over Ubuntu 10.04 Linux distribution is adopted. The network measurements were collected through a Wireshark Network Protocol Analyzer. Hardware measurements were obtained by instrumenting Tcpcrypt using Intel's VTune Performance Analyzer.

User-level Tcpcrypt exhibits slower performance than its kernel-level implementation [6] but it can be used in for hardware performance events measurement. Since native TCP is in the kernel-level, the performance comparisons are biased in favor of TCP. To remove this bias the case where both client and server communicate over Tcpcrypt against the case where the server communicates over TCP is compared. Also it is considered that the client communicates over Tcpcrypt having all the security functionality deactivated.

2.1 Overhead on Transmitted Data

A metric of particular interest for battery-powered mobile devices, whose energy consumption depends on the amount of transmitted and received data, is the overhead incurred by Tcpcrypt on the total volume of transmitted data. This overhead is due to two factors, extra bytes in the header of the packets for Tcpcrypt options, or extra bytes as a result of the encryption. In this evaluation the total transmission size in bytes of a file uploaded from a client to a server is measured, using six different file sizes. The data used for this evaluation were encapsulated in CDA (Clinical Document Architecture) format [7]. For each file transmission 50 iterations were executed and the average transmission size was calculated. The results are presented in Fig. 1. The overhead varies between 12.8 – 17.5% with the exception of the case where the file size is 897KB, for which the overhead is less than 1%. This overhead may become larger in a noisy channel due to the increased number of retransmissions.

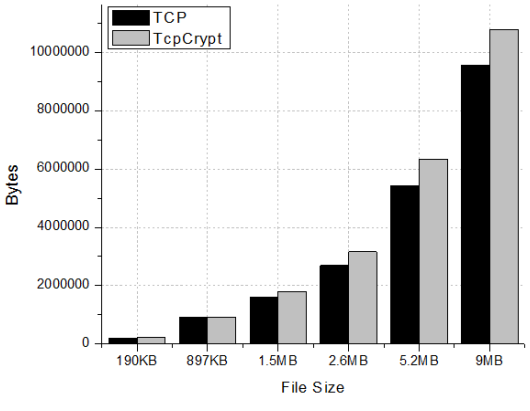

Fig. 1. Comparison of *transmitted bytes* with TCP and Tcpcrypt as the *File Size* increases

2.2 Overhead in CPU Instructions

A second metric related to the performance of mobile devices is the number of CPU instructions required. Given the limited resources of mobile clients, CPU utilization ideally should remain low. When encryption is disabled, the transmission of a 9MB file requires 1,036,000,000 instructions and spends 0.9 seconds of full CPU utilization. When encryption is enabled the number of executed instructions increases to 10,590,000,000, i.e. it requires an order of magnitude more instructions. The total time of full utilization increases to 8.3 seconds[1].

As shown in Fig. 2, the public-key connection initiation (the bursts during the first seconds) incurs the biggest cost. Tcpcrypt performs this operation in the client-side to reduce the stretch of the server's performance. After this initial phase the keys are cached and reused during further TCP communications, even for different TCP sessions. Thus, for a long-lived communication it is preferable to study the CPU utilization after the establishment of keys (second bursts in Fig. 2). When decryption is deactivated the data transmission requires 154,000,000 instructions, while when encryption is supported by both sides the number of instructions is increased to 1,036,000,000 (6.7 times more instructions). When a client communicates with only a limited number of servers, this initial phase does not impose a significant overhead. On the contrary, when the clients have to establish multiple connections with numerous different machines, the key generation incurs a prohibited large overhead. However, in a realistic scenario where a mobile device is engaged, in order to exchange health care data, a limited number of connections is required, rendering the use of Tcpcrypt an efficient solution. [8-10].

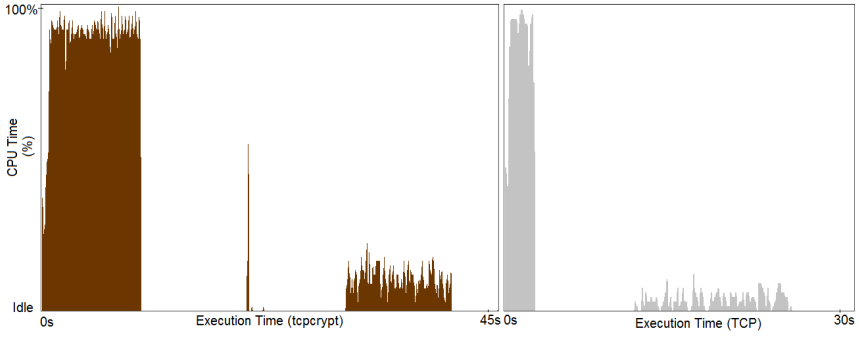

Fig. 2. CPU utilization and total duration for the transmission of data

2.3 Reducing Overhead through Fine-Grained Packet Handling

Availability is a critical performance metric for healthcare applications thus a system design that will enable a fine-grained handling of the TCP packets based on whether

[1] The CPU time depends on the CPU architecture and type and it is expected to differ for different processors.

the carried information is sensitive or not is proposed. Currently Tcpcrypt works as an on/off switch. If both ends support Tcpcrypt, encryption is activated for all the packets regardless of whether the information is confidential. This is unimportant for clients with spare CPU cycles and energy; however it would be more desirable to defer from encrypting packets with trivial information in resource-limited devices. Fig. 3 describes this functionality.

An application-specific module characterizes the packets criticality depending on the origin and type of the data (e.g. physiological sensors are marked as a sensitive source, temperature sensors as trivial). Then it passes the packets to the dual-stack TCP (Tcpcrypt/TCP) which operates simultaneously. The "Characterizer" module assigns five different levels of criticality to packets, 0-4, where 0 is trivial and 4 is highest confidentiality. If there is adequate energy all data regardless to the level assigned to them are passed to Tcpcrypt. As the battery discharges, only packets of a higher criticality level are passed to Tcpcrypt. Although this approach does not mitigate the initiation overhead it ensures low-cost CPU operations for long-lived sessions.

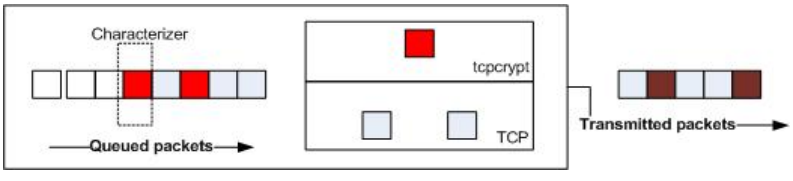

Fig. 3. The Characterizer addon to TCP dual stack. Only critical *(red/dark) packets* are passed to tcpcrypt. Two separate TCP sessions are maintained for each stack.

3 Discussion and Conclusions

When employing security mechanisms it is expected that certain overhead will be introduced. Although privacy and security requirements are of high importance it is imperative that they are used in a sensible manner especially in environments where resources are limited in terms of processing power, bandwidth, battery life etc. In such cases it may be preferable to downgrade security requirements in favor of the performance or the lifetime of the network.

In this paper, a comparison of Tcpcrypt against TCP was presented with a focus on health care applications which are sensitive in terms of integrity and confidentiality. In particular the overhead introduced was examined in terms of:

- increase in the volume of data to be transmitted
- CPU instructions

In the first case the increase occurred in different CDA file sizes was measured and was found roughly to be 12.8 – 17.5%. In the later case we have measured the CPU instructions and processing time overhead incurred. Enabling encryption results in an

increase of CPU processing time approximately 9 times up. While the instructions required for full encryption increase in a magnitude of almost 7 times.

As expected, Tcpcrypt is more demanding in both metrics and especially when it comes to CPU instructions. The increase in CPU operations results in increased power consumption that may have undesirable side effects in environments with limited resources. Furthermore non critical data may utilize Tcpcrypt depending on the availability of resources, and if for example a mobile device that is transmitting healthcare information of a patient and suffers from low battery situation TCP could be the most preferable solution.

Towards this direction, a methodology for classifying data in terms of criticality has been proposed. The goal is to minimize the overhead introduced by consuming valuable resources only for information that is considered critical. The implementation of Tcpcrypt, either standalone or in combination with the proposed classification scheme, in mobile devices would be valuable as the benefits provided can be maximized in such environments.

References

1. Ylonen, T., Lonvick, C.: The Secure Shell (SSH) Authentication Protocol, Network Working Group of the IETF, RFC 4252 (2006)
2. Rescorla, E.: HTTP Over TLS, Network Working Group of the IETF, RFC 2818 (2000)
3. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol, Network Working Group of the IETF RFC 5246 (2008)
4. Kent, S., Seo, K.: Security Architecture for the Internet Protocol Network Working Group of the IETF, RFC 4301 (2005)
5. Bittau, A., Boneh, D., Hamburg, M., Handley, M., Mazieres, D., Slack, Q.: Cryptographic Protection of TCP Streams (Tcpcrypt) draft-bittau-tcp-crypt-00.txt (2011)
6. Bittau, A., Hamburg, M., Handley, M., Mazieres, D., Boneh, D.: The Case for Ubiquitous Transport-Level Encryption. In: USENIX Security Symposium, Washington, DC (2010)
7. Alschuler, L., Dolin, R.H., Boyer, S., Beebe, C.: HL7 Clinical Document Architecture Framework, Release 1.0.ANSI-approved HL7 Standard (2000)
8. Widya, I., van Halteren, A., Jones, V., Bults, R., Konstantas, D., Vierhout, P., Peuscher, J.: Telematic Requirements for a Mobile and Wireless Healthcare System Derived from Enterprise Models. In: The Proceedings of 7th International Conference on Telecommunications, Croatia, pp. 527–534 (2003)
9. Boukerche, A., Yonglin, R.: A Secure Mobile Healthcare System using Trust-Based Multicast Scheme. IEEE Journal on Selected Areas in Communications, 387–399 (2009)
10. Nikolidakis, S., Georgakakis, E., Giotsas, V., Vergados, D.D., Douligeris, C.: A Secure Ubiquitous Healthcare System Based on IMS and the HL7 Standards. In: The Proceedings of the 3rd International Conference on Pervasive Technologies Related to Assistive Environments, Samos (2010)