Conway, Kiera

Points	95 / 100
Weight Achieved	9.5 / 10
see 4.b	

Homework 4
Track B

1

(60 points) Tcpcrypt is a TCP extension designed to make end-to-end encryption of TCP traffic the default, not the exception. To facilitate adoption tcpcrypt provides backwards compatibility with legacy TCP stacks and middle boxes. The paper, "The case for ubiquitous transport-level encryption", is available for you to go through all the details. You can also go to http://tcpcrypt.org/ and find more information about Tcpcrypt. Read the paper and answer the following questions:

A. (20 points) Tcpcrypt is a transport layer security protocol. As we discussed in the class, TLS is also a transport layer security protocol. Compare the differences between Tcpcrypt and TLS protocol (list at list three differences)

Although both TLS and Tcpcrypt are security protocols designed to secure network communications, they have distinct differences in their design focus, operational efficiency, and integration requirements. Specifically, TLS operates mainly on the application layer, where it secures communications between client and server applications through protocols such as HTTPS. It offers robust security through a detailed handshake process that includes key exchange, server authentication with X.509 certificates, and optionally client authentication [1]. As a result, despite providing high security, the complexity of TLS's handshake process and its dependency on certificates and public key infrastructure (PKI) lead to higher computational costs and administrative burdens. These requirements necessitate deliberate integration into applications, typically through libraries like OpenSSL, and are typically implemented only in scenarios deemed particularly sensitive.

However, Tcpcrypt redefines the notion of what is deemed 'sensitive' by advocating for a shift towards automatic and universal encryption to enhance internet security by default. By operating directly from the transport layer, Tcpcrypt focuses on minimizing the performance impact of encryption by reducing dependency on application-level support and the need for explicit configuration by end-users [2]. Unlike TLS, which requires active management including certificate installation, updates, and configuration of client and server software, Tcpcrypt simplifies these processes while offering more extensive encryption solutions.

Overall, Tcpcrypt's design prioritizes efficiency by streamlining key negotiations and leveraging efficient key management strategies. This approach enables Tcpcrypt to be deployed without disrupting existing network infrastructures and supports a higher volume of connections than TLS. Its lightweight encryption and simplified key exchange mechanism facilitate rapid key negotiation, even in environments experiencing high network traffic, making it a more efficient solution in scenarios requiring scalable encryption [2]. Therefore, as digital threats and networks continue to evolve, Tcpcrypt is paving the way towards a future where secure communications are standard practice, rather than the exception.

B. (20 points) What is the design consideration of Tcpcrypt protocol? Why is it necessary?

The Tcpcrypt protocol is designed to provide cryptographic protection to TCP streams in order to enhance the security of data transmission across networks. This design is necessary for not only mitigating various cybersecurity threats through encryption and integrity protection, including passive eavesdropping, desynchronization of sequence numbers, and injection of forged segments like RST (reset) packets, but also laying "a firm foundation upon which higher-level authentication mechanisms can build [2]."

One of the main goals of Tcpcrypt is to preserve the confidentiality and integrity of data transmitted over TCP segments. By embedding cryptographic protections directly within the TCP layer, Tcpcrypt ensures that data contents remain confidential and unaltered during transmission, even in the face of passive and active network attacks. One of the reasons this is possible is due to two new TCP options: CRYPT and MAC. The CRYPT option handles the negotiation and establishment of encryption parameters and keys between hosts, while the MAC option contains a message authentication code that confirms the integrity of each TCP segment [2].

Another significant advantage of Tcpcrypt's design is its relatively low computational overhead, especially on servers managing many connections simultaneously. Tcpcrypt reduces the cryptographic workload for each connection by allowing for the reuse of cryptographic parameters whenever possible. For example, once a key exchange has occurred between a specific client and server, the resulting session information is stored. This cached data can then expedite subsequent connections between the same parties by avoiding the full key negotiation process, thus significantly decreasing connection times and server load. Also, by integrating the initial key exchange into the TCP handshake, the number of messages exchanged during key negotiation are significantly reduced. These optimizations are necessary as they minimize the overhead often associated with other encryption protocols such as TLS, which require multiple exchanges to establish a secure connection. Tcpcrypt also uses ephemeral keys during session initiation, which improves security through forward secrecy to ensure that compromising long-term keys does not affect the security of past session keys [2]. This approach not only reinforces security but also streamlines the key negotiation process.

Overall, the need for Tcpcrypt stems from the increasing requirement for end-to-end security in internet communications. While traditional security measures such as TLS/SSL protect higher communication layers, they often leave the lower layers, such as TCP, vulnerable to attacks. Tcpcrypt addresses these gaps by embedding security features directly into TCP to ensure robust security for data across all application layer protocols used, thus safeguarding it against interception and manipulation.

C. (20 points) Tcpcrypt was originally proposed in 2010. What are the major challenges to adopt Tcpcrypt

There are a series of significant challenges that need to be addressed to encourage widespread adoption of Tcpcrypt, including complex key exchange processes, substantial performance overhead, and the necessity for compatibility with existing network infrastructure. The primary challenge in regards to tcpcrypt's performance relates to its computationally demanding key exchange process, which is essential for ensuring the previously mentioned forward secrecy. Since this process relies on public-key cryptography to generate a unique ephemeral key for each new session to ensure that compromising long-term keys does not affect past communications, this process is much more complex and resource-intensive than the simpler handshakes typical in standard TCP connections. Particularly, RSA decryption tasks, which are more demanding than encryptions, can create significant bottlenecks in environments with numerous short-lived sessions, such as web applications [2].

Additionally, the performance overhead linked to Tcpcrypt's key exchange mechanism could discourage its use in environments where delays are particularly harmful. To address this, Tcpcrypt offloads heavier decryption tasks to clients to help maintain server performance. In further efforts to minimize overhead, the protocol also employs session caching to reuse cryptographic states in subsequent connections between the same hosts, thus significantly reducing both the latency and computational overhead required to establish new sessions [2].

Another major hurdle is the integration of Tcpcrypt into existing network infrastructures. For widespread adoption to be feasible, the protocol must seamlessly interact with various network devices, including firewalls and load balancers, which may not inherently support encrypted TCP segments. Tcpcrypt combats this by embedding cryptographic operations within TCP's existing three-way handshake to enhance security without introducing additional network latency. However, this approach requires careful management of cryptographic data within the limited space of TCP options fields in order to ensure that crucial information fits without disrupting standard packet structures.

Adhering to this structure is vital as slight deviations can have unforeseen consequences. For example, handling network middleboxes that modify TCP packets based on preset rules is particularly challenging; these modifications can disrupt encrypted packets, thus necessitating that Tcpcrypt either avoids such disruptions or implements fallback strategies when interference occurs. Additionally, the complexity of managing TCP RST packets, which in Tcpcrypt must carry valid Message Authentication Codes (MACs) to verify their authenticity and integrity, poses a further challenge. Not all network configurations or devices may support or expect MACs on RST packets, and some might strip unknown options or data seen as extraneous, potentially leading to dropped packets or connection issues [2].

Overall, the widespread adoption of Tcpcrypt is contingent upon overcoming substantial technical challenges, including complex key exchange processes, notable performance overhead, and intricate integration issues with current network systems. Addressing these challenges is crucial to ensure that Tcpcrypt can provide security enhancements without complicating network operations, thereby facilitating more secure and widespread communications across various platforms.

2

(15 points) Figure 1 shows a diagram of how to retrieve public keys using a public key authority. In steps 3, 6, and 7, two nonce N1, N2 are used. Explain why N1, N2 are used in the protocol and what security service can be ensured by using N1, and N2 in the protocol. (Hints: think about non-repudiation service in OSI model)

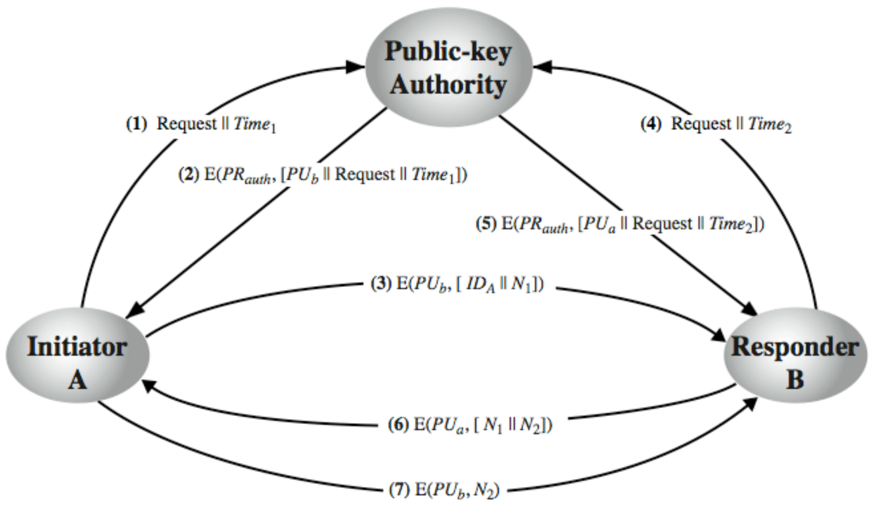
Public-key Authority
Initiator A
Responder B
(1) Request || $Time_1$
(2) E(PR_{auth}, [PU_b || Request || $Time_1$])
(3) E(PU_b, [ID_A || N_1])
(4) Request || $Time_2$
(5) E(PR_{auth}, [PU_a || Request || $Time_2$])
(6) E(PU_a, [N_1 || N_2])
(7) E(PU_b, N_2)

Figure 1. Retrieve Public Keys using a Public-key Authority

A. Explain why N_1, N_2 are used in the protocol:

In step 3, N_1 is included along with the identity of Initiator A in the message encrypted with Responder B's public key. The inclusion of N_1 is important because it acts as a marker of the session's initiation and ensures that any response from B that includes N_1 confirms B's receipt and processing of A's message. By acknowledging N_1 in a subsequent message, B is able to prove that it is present and actively engaged in the communication.

In step 6, Responder B responds to Initiator A by including both N_1, which was originally sent by Initiator A, and a new nonce, N_2. The retransmission of N_1 authenticates B to A and proves that

B received and decrypted A's initial message. Then, by introducing N_2, B contributes a way for A to respond with N_2 to confirm and validate communication.

For step 7, Initiator A sends a message back to Responder B which includes N_2 encrypted with B's public key. This step ensures that A has successfully received and decrypted B's prior message. Sending back N_2 not only verifies A's presence but also confirms a mutual authentication process, as it proves A’s active involvement and completion of the nonce exchange cycle.

Overall, the use of nonces N_1 and N_2 in the protocol serves multiple important functions. Since each nonce is unique to its session, by using them, both parties are able to ensure that their messages are current and no previous messages are being replayed by an attacker, which is essential in protecting against replay attacks. This uniqueness also binds each message to a specific session and participant, which means that even if a key were compromised, it would not be possible to use any intercepted messages in a different context. Secondly, nonces help authenticate the parties to each other. When a nonce generated by one party is signed or encrypted and sent back by another party, it ensures that the response is not pre-recorded and that the other party is indeed present and active at the time of the exchange. Then, by successfully processing and responding to the nonce, each party demonstrates their live participation in the communication and implicitly confirms receipt of the message.

B. What Security Service can be ensured by using N_1, and N_2 in the protocol:

The three primary security services that can be ensured by using N_1 and N_2 in the protocol are Authentication, Data Integrity, and Non-repudiation. The most direct service provided by the nonces is authentication. By including and correctly responding to nonces, both Initiator A and Responder B authenticate each other's identities. This process occurs as each party proves they have received, decrypted, and processed a message that could only have originated from the stated sender, due to the inclusion of a nonce known only within the context of their direct exchange. This can be seen in steps 6 and 7, where B and A authenticate each other by correctly handling the nonces they respectively generated.

Nonces can also ensure data integrity as the presence of a nonce in each communication step allows the recipient to verify that the message has not been tampered with and is indeed the message sent by the peer. If a message were tampered with, the nonce would not match what the recipient expects. Similarly, it provides a way to detect replay attacks since an old message replayed by an attacker would contain an outdated nonce.

Lastly, nonces also support aspects of non-repudiation as a party can not deny receipt of a message after it has engaged in communication with the other party. For example, Responder B cannot deny receiving A’s message in step 3 because step 6 included nonce N_1 , which was uniquely generated by Initiator A and included in the initial communication.

3

(15 points) Users A and B use the Diffie-Hellman key exchange technique with a common prime q = 71 and a primitive root α = 7. (Hints: refer to text book Figure 10.1 in Page 303)

A. (5 points) If user A has private key $X_{\text{A}} = 5$, what is A's public key Y_{A} ?

Given $q = 71$, $\alpha = 7$, $X_a = 5$, find Y_a

$$Y_a = \alpha^{X_a}\ mod\ q$$
$$Y_a = (7)^5\ mod\ 71$$
$$Y_a = 16807\ mod\ 71$$
$$Y_a = 51$$

B. (5 points) If user B has private key $X_{\text{B}} = 12$, what is B's public key Y_{B} ?

Given $q = 71$, $\alpha = 7$, $X_b = 12$, find Y_b

$$Y_b = \alpha^{X_b}\ mod\ q$$
$$Y_b = (7)^{12}\ mod\ 71$$
$$Y_b = (1.38e^{10})\ mod\ 71$$
$$Y_b = 4$$

C. (5 points) What is the shared secret key?

A computes $(Y_b)^{Xa}\ \ mod\ q = (\alpha^{Xb}\ mod\ q)^{Xa}\ mod\ q = \alpha^{Xb \times Xa}\ mod\ q$

$$(Y_b)^{Xa}\ \ mod\ q$$
$$(4)^5\ mod\ 71$$
$$1024\ mod\ 71$$
$$30$$

B computes $(Y_a)^{Xb}\ mod\ q = (\alpha^{Xa}\ \ mod\ q)^{Xb}\ mod\ q = \alpha^{Xa \times Xb}\ mod\ p$

$$(Y_a)^{Xb}\ mod\ q$$
$$(51)^{12}\ mod\ 71$$
$$30$$

Their shared key is 30.

4 *(10 points) Consider a one-way authentication technique based on asymmetric encryption:*

1.	$A \to B$	:	ID_A
2.	$B \to A$	:	$E(PU_a, \mathrm{R}_2)$
3.	$A \to B$	:	R_2

A. Explain the protocol

$$A \to B : ID_a$$

- Initiator A sends their identity ID_a to Responder B so that B can recognize who is trying to authenticate
- B needs this identity to retrieve A's public key from a trusted public key repository

$$B \to A : E(PU_a,\ R_2)$$

- Responder B then generates a challenge R_2 , or randomly generated piece of data, and encrypts it using the public key of Initiator A PU_a
- Since A is the only one who has the corresponding private key, they should be the only one able to decrypt this message. This verifies whether A is the owner of and has access to the private key.

$$A \to B : R_2$$

- Initiator A responds to B by sending back the decrypted challenge R_2 to prove that they successfully decrypted the message using their private key.
- A successful decryption proves that A possesses the corresponding private key associated with the public key PU_a that B used to encrypt R_2. Essentially, if A correctly sends back R_2, it confirms A's identity and completes the authentication process.
- B now knows that A is the owner of PU_a and is therefore the legitimate holder of the identity ID_a

B. What type of attack is this protocol susceptible to?

In this protocol, during Step 2, B sends a challenge R_2 encrypted with A's public key PU_a. A decrypts this challenge and sends it back in plaintext in Step 3. However, an attacker could capture the decrypted challenge R_2 sent to B by A. Since R_2 is sent in plaintext, the attacker can extract it and resend it later to B, pretending to be A. Since there are no nonces or timestamps in this protocol to verify if the correspondence is outdated, B would accept the old R_2 as valid and mistakenly authenticate the attacker as A. However, it is also important to note that the delays present in real-world asynchronous communication could make timestamps less effective for mitigating these types of attacks.

References

[1] "TLS and SSL," IBM Corporation, 27 03 2024. [Online]. Available: https://www.ibm.com/docs/en/zos/3.1.0?topic=protocols-tls-ssl.

[2] A. Bittau, M. Hamburg, M. Handley, D. Maziéres and D. Boneh, "The case for ubiquitous transport-level encryption," in *USENIX Security*, Washington, DC, 2010.