2009 Third International Conference on Network and System Security

Effective Attacks in the Tor Authentication Protocol

Yang ZHANG

State Key Laboratory of Networking and Switching Technology
Beijing University of Posts & Telecommunications
Beijing 100876, China
zhang_yang_own@yahoo.com.cn

Abstract—**As an anonymous Internet communication system Tor is popular and famous, being used by lots of users. The security of Tor is based on the authentication protocol. Although the Tor authentication protocol has been proved secure, this paper discovers its security vulnerability through its concurrency analysis, and shows it cannot be securely executed by multiple concurrent sessions. A new session-key exchange protocol for Tor is proposed to dispose of the security vulnerability, where a modular method is adopted to design a secure key exchange protocol in realistic world. Finally, the proposed protocol is proved secure in the UC (universally composable) model which defines conditions for a protocol to securely compose with other protocols in a concurrent environment.**

Keywords-Tor Authentication Protocol; Tor; Universally Composable Security; Onion Routing Protocol; Authenticator

I. INTRODUCTION*

As the world moves toward pervasive communication, anonymous networks will be used to protect user's privacy. In the works of [1,2,3,4], some anonymous Internet systems were proposed and designed. Formalizations and security discussions of those systems can be found in [5,6,7,8,9].

To date, the Tor (The Onion Router) anonymous communication system [1] is the most famous and practical distributed anonymous network in use [10], which is based on an onion routing protocol. There are other well-known anonymizing networks: JAP [11] and Freenet [12]. In order to protect users' privacy, Tor utilizes a number of nodes (also known as "onion routers" or "ORs") situated around the Internet, and a client builds a circuit by choosing a set of ORs. Before constructing onions, the client negotiates session keys with ORs in his chosen circuit by running the Tor authentication protocol. Because lots of users often access Tor concurrently, the security for multiple concurrent instances of the Tor authentication protocol plays a key role in efforts to provide anonymous communication services. I. Goldberg [9] proved the security of the Tor authentication protocol in 2006 when a single instance is running, which he called TAP. However, he did not discuss the security of the TAP protocol when it runs concurrently.

We discover the security vulnerability of the Tor authentication protocol during its concurrent run. When multiple instances of TAP are concurrently running, adversaries can trickily interleave the messages from these instances and make different keys between a client and an onion router in an instance without being perceived by them, which violates the original object of the TAP protocol.

The UC model is proposed by R. Canetti in 2001 [13], which defines conditions for a protocol to securely compose with other protocols in a concurrent environment. The works of [5,14,15,16] used this model to analyze the anonymity and concurrent compsobility of protocols. This paper adopts it to analyze the security of our protocols and calls an instance of protocols after a session.

We regard the following as the main contributions of this paper:

1) The TAP protocol is showed that it will results in inconsistency of session keys in two communication parties.
2) The TAP protocol is proved that it is not secure in the concurrent environment.
3) The TAP protocol is improved to be secure in

* Supported by the National Natural Science Foundation of China under Grant No. 60432010; National Grand Fundamental Research 973 Program of China under Grant No. 2007CB307103; Postdoctor Science Fund of China under No. 20090450335.

978-0-7695-3838-9/09 $26.00 © 2009 IEEE
DOI 10.1109/NSS.2009.94

the concurrent environment.

This paper is structured as follows. Section 2 gives a description on preliminaries. Section 3 describes the reason why the TAP protocol is insecure. The improved solution of the TAP protocol is given in section 4. Finally, a conclusion is drawn in Section 5.

II. PRELIMINARIES

A. The CK model

M. Bellare, R. Canetti, and H. Krawczyk [17] introduced the BCK security model in 1998 which allows researchers to adopt a modular approach to design and analyze protocols. This work was later extended by Canetti and Krawczyk [18] and referred to as the CK model.

The definition of security for key-exchange (KE) protocols is based on a game in the CK model. In this game, protocol π is modeled as a collection of n processes running at different parties $P_1,\cdots,P_n$ as well as an adversary. Each process is regarded as an interactive probabilistic polynomial-time (PPT) machine. Each instance of π is defined as a session, and each party may have multiple sessions running concurrently. The communications channel is controlled by the adversary S, also a PPT machine, which schedules and mediates all messages from sessions between the parties.

The CK model defines two adversarial models: *Authenticated-links adversarial Model (AM)* and *Unauthenticated-links adversarial Model (UM)*. The AM defines an idealized adversary that is not allowed to generate, inject, modify, replay and deliver messages of its choice except if the message is purported to come from a corrupted party. That is, an AM–adversary can only activate parties using messages from other parties in π. The UM is a more realistic model without the above restriction to adversaries. A UM–adversary can forge messages and deliver any messages of its choice.

A protocol that is secure in the AM can be converted into a secure protocol in the UM by applying an authenticator to it, i.e., the authenticator assures they can emulate each other. An authenticator is a protocol translator C that takes as input a protocol π and outputs a protocol $\pi' = C(\pi)$, with the property that if π is secure in the AM, then π' is secure in the UM because of π' emulating π. Authenticators can be constructed by applying a message transmission (MT) authenticator to each of the messages of the input protocol.

Two communicating parties P_i and P_j are said to have a *matching session* if they have sessions whose session-ids are identical and they recognize each other as their respective communicating partner for the same session. The adversary S may issue the query test-session (P_i,s). To respond to this query, a random bit $b \in_R \{0,1\}$ is selected. If $b=1$ then the session key is returned. Otherwise, return a random key chosen from the probability distribution of keys generated by the protocol. This query can only be issued to a session that has not been exposed.

The adversary S is allowed during the game to perform a test-session query to a party and session chosen by itself. S can only issue once the test-session query which will not be exposed either. S finally outputs a bit b' as its guess on whether the returned value is the session key or a random number. S wins the game if $b'=b$. The definition of security follows.

Definition 1. *A KE protocol π is called SK-secure in the UM (AM) if the following properties are satisfied for any UM-adversary (or AM-adversary).*

1. *If two uncorrupted parties complete matching sessions, then they both output the same key;*
2. *The probability that S guesses correctly the bit b is no more than 1/2 plus a negligible fraction in the security parameter.*

B. The UC model

In the UC security model [13], the network is asynchronous without guaranteed delivery of messages. The communication is public. The adversary is adaptive in corrupting parties and is only restricted to probabilistic polynomial time computation. The security requirements of a given tasks are captured via a set of instructions for a "trusted party" that obtains the inputs of the participants and provides them with the desired outputs. Informally, a protocol securely carries out a given task if running the protocol amounts to "emulating" an ideal process where the parties hand their inputs to a trusted party and obtain outputs from it. The algorithm run by the trusted party is

called an ideal functionality.

In order to allow the concurrent composition, an additional adversarial entity, called the environment is introduced. The environment generates the inputs to all parties, reads all outputs, and in addition interacts with the adversary in an arbitrary way throughout the computation. A protocol is said to securely realize a given ideal functionality if for any adversary there exists an "ideal-process adversary", such that no environment can tell whether it is interacting with the adversary and parties running the protocol, or with the ideal-process adversary and parties that interact with the ideal functionality. In its general form, the concurrent composition theorem basically says that if a protocol ρ securely realizes an ideal functionality F, then an execution of the composed protocol π^{ρ} "emulates" an execution of a protocol π in the F–hybrid model (i.e., replacing an invocation of the protocol ρ with a call to F). If the protocol π securely realizes some functionality G in the F–hybrid model, then π^{ρ} securely realizes G in the real-life model (i.e., π directly invokes the protocol ρ).

In order to define the ideal functionality of a key exchange [19], a special type of probabilistic interactive Turing machine is presented which is called a non-information oracle. Essentially, a non-information oracle N_o has the property that its local output is computationally independent from its communication with the outside world. The key exchange functionality F_{KE} is defined as follows.

Definition 2. *F_{KE} proceeds as follows running on security parameter k, with parties $P_1,\cdots,P_n$ and an* adversary *S. The functionality is parameterized by a non-information oracle N_o.*

1. *Upon receiving $(Establish\text{-}session,sid,P_i,P_j,role)$ from some party P_i, record the tuple $(sid,P_i,P_j,role)$, send this tuple to S. In addition, if there is a recorded tuple $(sid,P_j,P_i,role')$ then proceed as follows:*
   - *(a) If both P_i and P_j are uncorrupted then send (key,sid,P_i,P_j) to S and invoke N_o* with fresh random input. Whenever *N_o generates a message, send this message to S.* Whenever *S sends a message to N_o, forward this message* to *N_o. When N_o generates local output κ, send (key,sid) to S. When receiving (ok,sid,l) from S where $(l=i,or,l=j)$, send (key,sid,κ) to P_l.*
   - *(b) If either P_i or P_j is corrupted, send $(Choose\text{-}value,sid,P_i,P_j)$* to *$S$. Receiving a value κ from S,* send *(key,sid,κ) to P_i and P_j.*
2. *If S corrupts P_l $(l=i,or,l=j)$ before the message addressed to P_l in step 1 is sent, then provide S with the internal state of N_o. Otherwise, do nothing.*

According to the conclusion of [19], UC-security is equivalent to SK-security. That is, **definition 1** is equivalent to **definition 2.** Therefore, we often adopt definition 1 to analyze the concurrent composability of session-key exchange protocols. The UC-secure channel functionality F_{SC} can be referred to the works of [18,19].

C. The DDH assumption

Definition 3. Decisional Diffie-Hellman assumption (DDH). *Let G be a multiplicative group of prime order p, g be a generator of the group G, and the discrete logarithm problem be hard in the group G. Then the probability distributions of quintuples*

$$Q_0=\{(p,g,g^x,g^y,g^{xy}):x,y\overset{R}{\leftarrow}Z_p\}$$

$$Q_1=\{(p,g,g^x,g^y,g^z):x,y,z\overset{R}{\leftarrow}Z_p\}$$

are computationally indistinguishable.

III. THE SECURITY VULNERABILITY OF TAP

A. The TAP protocol

Assume the user P authenticates the onion router A and negotiates a session key with A. The underlying building blocks are as follows:

1) There is a trusted PKI that allows P to obtain A's public encryption key PK_A. Let E_{PK_A} be public-key encryption using A's public key, and let D_{PK_A} be the corresponding decryption using A's private key, where the public key encryption scheme is IND-CPA (indistinguishability under chosen plaintext attacks).
2) p is a prime such that $q=(p-1)/2$ is also prime,

and g is a generator of the subgroup of Z_p^* of order q.

3) H is a cryptographic hash function, which is modeled by a random oracle.

The TAP protocol between P and A is as follows:

1) $P \to A$: $E_{PK_A}(g^x)$, where x is a random number picked by P.

2) $A \to P$: $g^y, H(g^{xy} | "handshake")$, where y is a random number picked by A, and $g^{xy} | "handshake"$ is the concatenation of g^{xy} and "$handshake$".

Preliminary analysis with the NRL protocol analyzer [20] shows this protocol to be secure under the traditional Dolev-Yao model. Ian Goldberg directly proved in his paper [9] that it is secure under the IND-CPA assumption of the public key encryption scheme. However, this paper shows it is not secure when multiple sessions are executed concurrently.

B. Analysis of TAP

Assume the adversary S has corrupted and controlled one onion router R with which the user P communicates and negotiates a session key (session $s1$). In addition, P also negotiates a session key with another onion router A (session $s2$). S interleaves the messages from the sessions $s1$ and $s2$ to establish different session keys in P and A without being perceived by P or A. Therefore, the original object of the TAP protocol is violated. We also prove in the UM that the TAP protocol is not secure and cannot be securely executed concurrently.

Theorem 1. *If multiple sessions of TAP are running, two parties in one session probably negotiate different session keys even though they are uncorrupted.*

Proof. The adversary S interleaves the messages in $P \leftrightarrow R$ ($s1$), $P \leftrightarrow A$ ($s2$) and $R \leftrightarrow A$ ($s3$) to attack TKE as follows:

$P \to R$ ($s1.1$): $E_{PK_R}(g^x)$, x is a random number picked by P and R has g^x.

$P \to R(A)$ ($s2.1$): $E_{PK_A}(g^{x'})$, x' is a random number picked by P, $E_{PK_A}(g^{x'})$ is intercepted by R, and R has $E_{PK_A}(g^{x'})$.

$R \to A$ ($s3.1$): $E_{PK_A}(g^{x'})$, R replaces the source address of $E_{PK_A}(g^{x'})$ with itself.

$R(P) \to A$ ($s2.2$): $E_{PK_A}(g^x)$, R impersonates P and encrypts g^x with the public key of A.

$A \to R$ ($s3.2$): $g^z, H(g^{x'z} | "handshake")$, z is a random number picked by A.

$A \to R(P)$ ($s2.3$): $g^y, H(g^{xy} | "handshake")$, y is a random number picked by A, $g^y, H(g^{xy} | "handshake")$ is intercepted by R, and A has a session key g^{xy}.

$R(A) \to P$ ($s2.4$): $g^z, H(g^{x'z} | "handshake")$, R impersonates A and P has a session key $g^{x'z}$.

$R \to P$ ($s1.2$): $g^y, H(g^{xy} | "handshake")$.

Finally, in the session $P \leftrightarrow A$ ($s2$), P has a session key $g^{x'z}$ and A has a session key g^{xy}. The session keys in P and A are different, which violates the original object of the TKE protocol.

Theorem 2. *The TAP protocol is not SK-secure in the UM according to* ***Definition*** **1**.

Proof. We prove that the second requirement of **Definition 1** is not satisfied by TKE. That is, there is a KE-adversary S in the UM against the TKE protocol that has a non-negligible advantage in guessing correctly whether the response to a test-query is real or random**.** Assume the session sid is chosen as a test session by S. The real session key from sid is k and the value returned to S is k' in the game. S first computes

as follows:

$$m = k' \,|\, "handshake"$$

Then, S accesses to the random oracle H with m as input, and obtains h responded by H.

Finally, S compares h with $l = H(g^{xy} \,|\, "handshake")$, the second item in the second message of the TKE protocol, which can be directly read by S. S draws a conclusion as follows:

If $h = l$, then $b = 1$, else $b = 0$.

Therefore, the probability that S guesses correctly the bit b is 1 which is far greater than 1/2. In other words, the TAP protocol is not SK-secure in the UM.

According to the conclusion in Section 2.2 that UC-security is equivalent to SK-security, the TAP protocol is not UC-secure either. **Theorem 3** is induced by **Theorem 1** and **Theorem 2.**

Theorem 3. *The TAP protocol is not UC-secure and cannot be securely executed concurrently.*

IV. Improvement of TAP

We adopt the modular approach in the CK model [17, 18] to improve the TAP protocol. Firstly, the Diffie-Hellman key exchange is reserved, which is SK-secure in the AM [18]. Secondly, a signature-authenticator [17] is adopted to compile the SK-secure Diffie-Hellman key exchange to the SK-secure protocol in the UM. Because the public key of any client is not known by ORs in the Tor system, a one-time public key is generated for each OR in a circuit by the client, which is attached to the first message in TAP. The one-time public key can be regarded as a temporal identity and need not to be bound to the real identity of the client because ORs just serve anonymous clients. Finally, the last message in the improved TAP (named AKE) can be attached to the first message in the next AKE session with a subsequent OR, which probably reduces the rounds of interaction. In order to achieve further reduction of rounds of interaction, a time-signature-authenticator [21] may be appropriate.

The AKE protocol between a client P_i and an onion router P_j is as follows, which adopts a signature scheme $\prod_{Sig}(Gen, Sign, Verify)$ to construct a signature-authenticator.

Initial information: *Prime p, G of order p, and g of the G where the discrete logarithm problem is hard in G.*

Step 1: *On input (P_i, P_j, s) where s is a session id, P_i generates a one-time public/private key pair (PK, SK), chooses $x \xleftarrow{R} Z_p$, and sends $(P_i, s, PK, \alpha = g^x)$ to P_j.*

Step 2: *Upon receipt of (P_i, s, PK, α), P_j chooses $y \xleftarrow{R} Z_p$, computes $\beta = g^y$ and $\sigma = Sign_{SK_j}(P_j, s, \beta, PK, \alpha, P_i)$, sends the message (P_j, s, β, σ) to P_i. It also computes the session key $K = \alpha^y$ and erases y.*

Step 3: *Upon receipt of (P_j, s, β, σ), P_i verifies the signature and the correctness of the values included in the signature. If the verification succeeds, then P_i computes $\sigma' = Sign_{SK_i}(P_i, s, \alpha, \beta, P_j)$, sends the message (P_i, s, σ') to P_j. It also computes the session key $K' = \beta^x$, erases x, and outputs K'.*

Step 4: *Upon receipt of (P_i, s, σ'), P_j verifies the signature and the included values. If the verification succeeds, it outputs the session key K.*

According to **Claim 19** of [19], the AKE protocol is UC-secure, which is described as follows:

Theorem 4. *If the signature scheme $\prod_{Sig}$ is secure under adaptive chosen message attacks and the DDH assumption is true, the AKE protocol is UC-secure and can be concurrently composed with other protocols.*

V. Conclusions

This paper illustrates with an example how to succeed in attacking TAP by interleaving messages from multiple instances of TAP. At the same time, we prove the TAP protocol is not SK-secure and cannot be securely executed concurrently. In order to solve above problems, this paper adopts the modular approach to improve the protocol. Firstly, the SK-secure Diffie-Hellman key exchange is reserved. Secondly, a signature-authenticator is adopted to translate the SK-secure protocol in the AM to the SK-secure protocol in the UM. Finally, the improved protocol, AKE, is proved to be UC-secure.

References

[1] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In Proceedings of the 13th USENIX Security Symposium, August 2004.

[2] M. J. Freedman and R. Morris. Tarzan: A Peer-to-Peer Anonymizing Network Layer. In Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002), Washington, DC, November 2002.

[3] M. Reed, P. Syverson, and D. Goldschlag. Anonymous Connections and Onion Routing. IEEE Journal on Selected Areas in Communications, 16(4): pp. 494, May 1998.

[4] M. Rennhard and B. Plattner. Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2002), Washington, DC,USA, November 2002.

[5] J. Camenisch and A. Lysyanskaya. A Formal Treatment of Onion Routing. In Advances in Cryptology-CRYPTO 2005, Lecture Notes in Computer Science 3621, pp. 169-187. Springer-Verlag, August 2005.

[6] S. Mauw, J. Verschuren, and E. de Vink. A Formalization of Anonymity and Onion Routing. In ESORICS 2004, Lecture Notes in Computer Science 3193, pp. 109-124. Springer-Verlag, September 2004.

[7] B. Moller. Provably Secure Public-Key Encryption for Length-Preserving Chaumian Mixes. In CT-RSA 2003, Lecture Notes in Computer Science 2612. Springer-Verlag, April 2003.

[8] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr. Towards an Analysis of Onion Routing Security. In Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability, Lecture Notes in Computer Science 2009, pp. 96-114. Springer-Verlag, July 2000.

[9] I. Goldberg. On the Security of the Tor Authentication Protocol. In Proceedings of the Sixth Workshop on Privacy Enhancing Technologies (PET 2006), Lecture Notes in Computer Science 4258, pp. 316-331. Springer-Verlag, June 2006.

[10] The Tor Project. http:// www.torproject.org.

[11] O. Berthold, H. Federrath, and S. Kopsell. Web MIXes: A system for anonymous and unobservable Internet access. In H. Federrath, editor, Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability, pp. 115-129. Springer-Verlag, LNCS 2009, July 2000.

[12] I. Clarke, O. Sandberg, B. Wiley, and W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In Proceed-ings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability, pp. 46-66, July 2000.

[13] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In: Proc. IEEE Symp. on Foundations of Computer Science (FOCS 2001), IEEE Press, pp. 136-145, 2001.

[14] D. Wikstrom. A universally composable mix-net. In: Proc. of the Theory of Cryptography Conf. LNCS 2951, Springer-Verlag, pp. 317-335, 2004.

[15] M. Burmester, T. van Le, B. de Medeiros. Provably secure ubiquitous systems: Universally composable RFID authentication protocols. http://eprint.iacr.org/2006/131, 2006.

[16] G. Ateniese, J. Camenisch, B. DE Medeiros. Untraceable RFID tags via insubvertible encryption. In: Proc. ACM Conf. on Computer and Communication Security (ACM CCS 2005). NY, USA: ACM Press, pp. 92-101, 2005.

[17] M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In Proc. of the 30th Annual Symp. on the Theory of Computing, pp. 419-428, 1998.

[18] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. Advances in Cryptography, 2001, Vol.2045 of LNCS 2045, pp. 453-374, 2001.

[19] R. Canetti and H. Krawczyk. Universally composable notions of key exchange and secure channels. In Advances in Crypotology (EUROCRYPT 2002), pp. 337-351, 2002.

[20] C. Meadows. The NRL protocol analyzer: An overview. Journal of Logic Programming, 26(2): pp. 113-131, 1996.

[21] Y.S.T. Tin, H. Vasanta, C. Boyd, J.M.G. Nieto. Protocols with security proofs for mobile application. In: Proceedings of the ACISP2004, pp. 358-369, 2004.