On the Security of the 
Tor Authentication Protocol

Ian Goldberg

David R. Cheriton School of Computer Science, University of Waterloo, 200 University Ave W, Waterloo, ON N2L 3G1 iang@cs.uwaterloo.ca

Abstract. Tor is a popular anonymous Internet communication system, used by an estimated 250,000 users to anonymously exchange over five terabytes of data per day. The security of Tor depends on properly authenticating nodes to clients, but Tor uses a custom protocol, rather than an established one, to perform this authentication. In this paper, we provide a formal proof of security of this protocol, in the random oracle model, under reasonable cryptographic assumptions.

1 Introduction

The Tor anonymous communication system [11] is used by an estimated 250,000 users worldwide [15] to protect the privacy of their Internet communications. Users can maintain their anonymity with Tor while taking advantage of many Internet services, including web browsing and publishing, instant messaging, and ssh.

In order to protect users’ privacy, Tor utilizes a number of nodes (also known as “onion routers” or “ORs”) situated around the Internet. A client (an Internet user, whom we will call Alice, who does not necessarily run a node herself) builds a circuit through the network as follows:

- Alice picks a Tor node, \( n_1 \), and establishes an encrypted communication channel with it.
- Alice picks a second Tor node, \( n_2 \), and, over the previously established channel, instructs \( n_1 \) to connect to \( n_2 \). Alice then establishes an encrypted communication channel with \( n_2 \), tunneled within the existing channel to \( n_1 \).
- Alice picks a third Tor node, \( n_3 \), and, over the previously established channel, instructs \( n_2 \) to connect to \( n_3 \). Alice then establishes an encrypted communication channel with \( n_3 \), tunneled within the existing channel to \( n_2 \).
- and so on, for as many steps as she likes.

The security of the Tor system derives in part from the fact that the various nodes in the circuit are operated in different administrative domains; if one party had access to the internal state of all of the nodes in Alice’s circuit, he could easily compromise Alice’s anonymity.
For this reason, it is important that Alice be assured that her communications with the various nodes be authenticated: if Mallory (a malicious man-in-the-middle) operated (or compromised) any single node $n_i$, then, without authentication, he could simulate all subsequent nodes $n_{i+1}, n_{i+2}, \ldots$ in Alice’s circuit. If Alice were unlucky enough to pick Mallory’s node as her $n_1$, he would be able to control her entire circuit.

Therefore, at each step, Alice (a) establishes a shared secret with a node, and (b) verifies that node’s identity, so that it cannot be impersonated. Note that Alice’s identity is never authenticated; she operates anonymously.

Tor uses a new protocol to achieve this, which we call the Tor Authentication Protocol (TAP) \cite{10}. TAP is not an established authentication protocol, however, and its first deployment had at least one serious weakness \cite{9}. In this paper, we analyze the (updated) TAP, and give a formal proof of security in the random oracle model \cite{3}. This formal proof provides confidence that there are no similar weaknesses remaining in the protocol.

## 2 The Tor Authentication Protocol

We will first describe TAP in abstract terms. TAP is built from the following pieces:

- There is a trusted PKI that allows Alice to determine each node’s public encryption key. Let $\mathcal{E}_B$ be public-key encryption using $B$’s public key, and let $\mathcal{D}_B$ be the corresponding decryption using $B$’s private key.
- $p$ is a prime such that $q = \frac{p-1}{2}$ is also prime, and $g$ is a generator of the subgroup of $\mathbb{Z}_p^*$ of order $q$. $l_x$ is an exponent length; when a “random exponent” is required, select an $l_x$-bit value uniformly from the interval $[1, \min(q, 2^{l_x}) - 1]$.
- $f$ is a hash function, which we will model by a random oracle, taking as input elements of $\mathbb{Z}_p$, and outputting bit strings of length $l_f$.

The abstract protocol is as follows:

1. Alice selects a node to add to her circuit. Let us suppose she selects Bob ($B$).
2. Alice picks a random exponent $x$, and computes $g^x$ (all exponentiations will be assumed to be mod $p$, and the least nonnegative representative will always be used).
3. Alice sends $c = \mathcal{E}_B(g^x)$ to Bob.
4. Bob computes $m = \mathcal{D}_B(c)$, checks that $1 < m < p - 1$, picks a random exponent $y$, and computes $a = g^y$ and $b = f(m^y)$.
5. Bob sends $(a, b)$ to Alice.
6. Alice checks that $1 < a < p - 1$ and that $b = f(a^x)$ \footnote{The error corrected in \cite{9} was that Alice neglected to check that $1 < a < p - 1$. This allowed Mallory to ignore Alice’s first message, reply with $(1, f(1))$, and use the “shared secret” of 1 to read Alice’s subsequent messages, pretending to be Bob.}
7. If the checks are successful, Alice accepts Bob’s authentication, and they use $a^x = m^y$ as a shared secret in order to communicate privately.