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 of 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) [10]. TAP is not an established authentication protocol, however, and its first deployment had at least one serious weakness [9]. In this paper, we analyze the (updated) TAP, and give a formal proof of security in the random oracle model [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 $\mathcal{E}_{\mathcal{B}}$ be public-key encryption using $\mathcal{B}$'s public key, and let $\mathcal{D}_{\mathcal{B}}$ be the corresponding decryption using $\mathcal{B}$'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 ($\mathcal{B}$).
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 $c = \mathcal{E}_{\mathcal{B}}(g^x)$ to Bob.
4. Bob computes $m = \mathcal{D}_{\mathcal{B}}(c)$, 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 $b = f(a^x)$.[1]
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.

Note that in step 4, it is possible that $m \leq 1$ or $m \geq p-1$ or even that $\mathcal{D}_{\mathcal{B}}(c) = \perp$; i.e. c is not a valid ciphertext. In these cases, Bob aborts the protocol.

Remember that all of the communication in TAP, other than that between Alice and the first node, is visible to, and modifiable by, the previous node in the circuit. We will assume this node is malicious, and denote it by Mallory ($\mathcal{M}$).

3 Formalization

In this section, we formally define what we mean by the *security* of TAP.

We begin by formally defining, in the usual way, a public-key encryption system (in the random oracle model) Π as a triple $(\mathcal{K}, \mathcal{E}^H, \mathcal{D}^H)$ of algorithms:

- $\mathcal{K}$ is the *key generation algorithm*. It takes as input a security parameter, k, in unary notation, and outputs a pair $(\mathrm{pk}, \mathrm{sk})$. pk is the public key, and sk is the private key. $\mathcal{K}$ is a polynomial-time randomized algorithm. If additionally, there is a polynomial-time algorithm κ that inputs pk, and outputs the value of k used to generate it, we say that Π is *k-aware*.[2]
- $\mathcal{E}^H$ is the *encryption algorithm*. It has access to a random oracle H, takes as input a public key output by $\mathcal{K}$ and a plaintext message m, and outputs a ciphertext c. $\mathcal{E}^H$ is also a polynomial-time randomized algorithm, so there are many possible outputs c for the same inputs pk and m.
- $\mathcal{D}^H$ is the *decryption algorithm*. It has access to the same random oracle H, takes as input a private key output by $\mathcal{K}$ and a ciphertext c, and outputs either a plaintext message m, or else a special symbol $\perp$, indicating an invalid input. $\mathcal{D}^H$ is also polynomial-time, but is deterministic.

We of course require that, for any (pk, sk) output by $\mathcal{K}(1^k)$, any plaintext message m in the domain of $\mathcal{E}^H_{\mathrm{pk}}$ (which may depend on pk), and any c output by $\mathcal{E}^H_{\mathrm{pk}}(m)$, it must be the case that $\mathcal{D}^H_{\mathrm{sk}}(c) = m$. Note that we indicate the keys as subscripts to $\mathcal{E}^H$ and $\mathcal{D}^H$.

[1] The error corrected in [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.

[2] Almost every reasonable public-key encryption system is k-aware. k-awareness is an easy-to-verify technical condition that will prevent the use of certain pathological systems in section 6.

Next we define a *group parameter generator*. This is a function $\mathcal{G}$, which is possibly, but not necessarily, randomized. $\mathcal{G}$ takes as input a security parameter k, again in unary notation, and outputs a pair (p, g) such that:

- p is prime
- $q = \frac{p-1}{2}$ is also prime
- g is a generator of the subgroup of $\mathbb{Z}_p^*$ of order q
- the length of p, in bits, is $\Omega(k)$, and polynomial in k

Finally, we recall that a function $\epsilon(k)$ is *negligible* with respect to k if for every constant $c \geq 0$, there exists an integer k_c such that $\epsilon(k) \leq k^{-c}$ for all $k \geq k_c$. Throughout this paper, the term "negligible" by itself will mean "negligible with respect to the security parameter k".

Definition 1. *For a given public-key encryption system Π, and a given group parameter generator $\mathcal{G}$, we say TAP is $(\Pi, \mathcal{G})$-**insecure** if there exists a polynomial-time randomized algorithm $\mathcal{M}^{f,H,\mathcal{N}}$ such that, for a random output $(\mathrm{pk}, \mathrm{sk})$ of $\mathcal{K}(1^k)$, a (possibly random) output (p, g) of $\mathcal{G}(1^k)$, and a random exponent x, with non-negligible probability, $\mathcal{M}^{f,H,\mathcal{N}}(\mathrm{pk}, p, g, \mathcal{E}^H_{\mathrm{pk}}(g^x)) = (a, a^x)$, for some $1 < a < p - 1$.*

*If no such algorithm exists, we say TAP is $(\Pi, \mathcal{G})$-**secure**.*

As the notation would suggest, $\mathcal{M}^{f,H,\mathcal{N}}$ has access to three oracles: the random oracles f and H, and a *node oracle* $\mathcal{N}$. The node oracle is one which emulates the behaviour of the above Tor node Bob: given an input c, it outputs a pair $\left(g^y, f\left(\mathcal{D}^H_{\mathrm{sk}}(c)^y\right)\right)$ for an exponent y chosen freshly at random on each invocation.[3]

It is easy to see how, if such an algorithm exists, Mallory could compromise the security of Tor: when Alice asks Mallory to extend her circuit to Bob, Alice will choose an x and give him $\mathcal{E}_{\mathcal{B}}(g^x)$. Mallory will run $\mathcal{M}^{f,H,\mathcal{N}}$ on that value, Bob's public key, and the group parameters, replacing every call to $\mathcal{N}$ by contacting Bob, sending c, and receiving Bob's output $(g^y, f(\mathcal{D}_{\mathcal{B}}(c)^y))$ for an exponent y chosen freshly at random each time. Mallory takes the output (a, a^x), with $1 < a < p - 1$, and returns $(a, f(a^x))$ to Alice. Now Alice will send messages protected with the shared secret a^x, thinking that only Bob can read them. But Mallory knows this value, and Tor's security is compromised.

Broadly, there are two main strategies Mallory could use to construct such an algorithm. First, he could perform an attack on $\mathcal{E}_{\mathcal{B}}(g^x)$ to try to recover g^x; in this case, he can pick a random r and output $(g^r, (g^x)^r)$. Second, Mallory could try to construct a "master" (a, a^x) pair that works despite his not knowing g^x. It was this latter strategy that was exploited in the previous version of the Tor protocol, where the restriction $1 < a < p - 1$ was absent. The purpose of this paper is to show that Mallory has no way to succeed using either of these strategies, or indeed any other strategy.

[3] So long as $\mathcal{D}^H_{\mathrm{sk}}(c) \neq \perp$ and $1 < \mathcal{D}_{\mathrm{sk}}(c) < p - 1$. Otherwise $\mathcal{N}$ returns $\perp$.

4 IND-CPA

We next recall the definition of IND-CPA (indistinguishability in a chosen plaintext attack) [1]:

Definition 2. *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme. For a pair of randomized algorithms $A = (A_1, A_2)$, define the* **advantage** *of A to be*

$$\mathsf{Adv}^{\mathrm{ind-cpa}}_{A,\Pi}(k) = |2 \cdot \Pr\,[\ (\mathrm{pk}, \mathrm{sk}) \leftarrow \mathcal{K}\left(1^k\right); (m_0, m_1, \sigma) \leftarrow A_1\,(\mathrm{pk}); b \leftarrow \{0,1\}; \\ y \leftarrow \mathcal{E}_{\mathrm{pk}}\,(m_b) : A_2\,(m_0, m_1, \sigma, y) = b] - 1|\ .$$

We additionally require that the outputs m_0 and m_1 of A_1 be of the same length.

Definition 3. *If, for a given public-key encryption scheme Π, any pair of polynomial-time randomized algorithms A has advantage $\mathsf{Adv}^{\mathrm{ind-cpa}}_{A,\Pi}(k)$ negligible in k, we say Π is* **IND-CPA**.

Informally, we say Π is IND-CPA if there is no way for a polynomial-time adversary to win the following game (against a *tester*) non-negligibly more often than half the time: [4]

- (Key generation:) The tester generates a public/private key pair.
- (Message generation:) The adversary is told the public key, and picks two messages m_0 and m_1 of the same length.
- (Challenge:) The tester picks a bit b at random, and produces y, an encryption of m_b using the public key.
- (Response:) The adversary is given y, and tries figure out the value of b.

The adversary can keep state (symbolized by σ in the formalization) between the message generation and the response phases.

5 Reaction Resistance

In this section, we introduce the concept of *reaction resistance*, which is similar to plaintext awareness [1], but weaker. Therefore, we first review the latter. Informally, a public-key encryption system is said to be *plaintext aware* if there is no way to produce a valid ciphertext (one which does not decrypt to $\perp$) without knowing the corresponding plaintext. Even if you observe some valid ciphertexts (say, by intercepting messages), you should be unable to modify them to produce a new ciphertext whose decryption you do not know.

[4] The adversary can trivially win half the time by simply picking an answer at random.

We formalize this by saying that an IND-CPA public-key encryption system is plaintext aware if there is an algorithm K, known as a *knowledge extractor*. K is given the public key pk, a list C of *observed ciphertexts* (for which the plaintext is not necessarily known), a *challenge ciphertext* $y \notin C$, generated by some adversary, and the list η of all of the random oracle queries and responses used by the adversary to construct y. K must then output the correct decryption of y, except with negligible probability.

Plaintext awareness is a very strong property for a public-key encryption system to have; for example, any system which is plaintext aware is also resistant to adaptive chosen ciphertext attacks. We define the weaker property of reaction resistance, which allows the adversary to construct certain valid ciphertexts for which it doesn't know the corresponding plaintext. To do this, we remove the requirements that the system be IND-CPA and that $y \notin C$, and also allow the knowledge extractor to do any of the following:

- output (plain, v) where v is the decryption of y, as in the plaintext awareness case (v is allowed to be $\perp$)
- output (match, i), claiming that the decryption of y is the same as that of C_i (this can be used when $y \in C$, for example)
- output (guess), indicating that the only way for an adversary to know the decryption of y is to guess it (note that the extractor is given the transcript of the adversary's calls to the random oracle, so it would know how y was generated)

Formally:

Let $\Pi = (\mathcal{K}, \mathcal{E}^H, \mathcal{D}^H)$ be a public-key encryption system where the encryption and decryption algorithms have access to a random oracle H.

Let B^H be an adversary that is given a public key pk output by $\mathcal{K}$ and a list of ciphertexts C, and outputs a tuple (η, y, m), where:

- η is the list of queries and results that B^H made to the random oracle H
- y is a *challenge ciphertext*
- m is a *guessed plaintext*

Definition 4. *Π is* **reaction resistant (RR)** *if there exists a knowledge extractor K, which, for any such adversary B^H, has the property that*

$$K(\eta, C, y, \mathrm{pk}) = \begin{cases} (\mathsf{plain}, v) & \Rightarrow \mathcal{D}^H_{\mathrm{sk}}(y) = v \\ (\mathsf{match}, i) & \Rightarrow \mathcal{D}^H_{\mathrm{sk}}(y) = \mathcal{D}^H_{\mathrm{sk}}(C_i) \\ (\mathsf{guess}) & \Rightarrow \mathcal{D}^H_{\mathrm{sk}}(y) \neq m \wedge \forall i : \mathcal{D}^H_{\mathrm{sk}}(y) \neq \mathcal{D}^H_{\mathrm{sk}}(C_i) \end{cases}$$

for any $(\mathrm{pk}, \mathrm{sk})$ *output by $\mathcal{K}$, any list of ciphertexts C (created by using $\mathcal{E}^H_{\mathrm{pk}}$ to encrypt plaintexts selected from some distribution), and any (η, y, m) output by $B^H(\mathrm{pk}, C)$, except with negligible probability.*

We note that this definition of RR is in the random oracle model, and follows the definition of plaintext awareness from [1]. The methods used in [2] could be used to produce a definition of RR in the standard model, but this is not necessary for our purposes.

Reaction resistance is so named because it is the property a cryptosystem needs in order to prevent reaction attacks [13], such as those against Atjai-Dwork [13], NTRU [14], and PKCS#1-v1.5 [5]. In these attacks, the adversary sends chosen ciphertexts (typically modified versions of intercepted ciphertexts) to one of the participants in the protocol, and watches her reaction in order to determine whether the ciphertext decrypted to something sensible. This information can be enough for the adversary to determine the original plaintext, or sometimes the secret key.

Finally, we define the weaker notion of RR1 (reaction resistance with a single observed ciphertext):

Definition 5. *Π is* **RR1** *if there exists a knowledge extractor K that satisfies the conditions of Definition 4, but which may also assume that its second parameter, C, is a list consisting of exactly one ciphertext.*

6 Security Reduction

In this section, we provide a reduction from the security of TAP to the security of the underlying public-key encryption system. We start by defining an x-power pair, and the $\mathcal{G}$-restriction of a public-key encryption system. The latter is just a slight modification to the original system that additionally checks that decrypted values are integers from some particular interval.

Definition 6. *For a fixed exponent x, an* **x-power pair** *is a pair (α, α^x) such that $1 < \alpha < p - 1$.*

Note: It will be important in section 6.2 that an algorithm that knows no information about x, save that is a random exponent, be able to create an x-power pair only with negligible probability. Since $1 < \alpha < p - 1$, and each member of this interval has order either $\frac{p-1}{2}$ or $p - 1$, this is trivially true. However, if the restriction on α were not present, an attacker could choose an element of low order for α, and easily create an x-power pair, so the proof of Theorem 1 would not go through. This was the problem in the earlier version of the protocol.

Definition 7. *If Π is a public-key encryption system, and $\mathcal{G}$ is a group parameter generator, then the* **$\mathcal{G}$-restriction** *of Π, denoted $\Pi^{\mathcal{G}}$, is a public-key encryption system $(\mathcal{K}^*, \mathcal{E}^*, \mathcal{D}^*)$, where*

- $\mathcal{K}^*(1^k) = ((\mathrm{pk}, p, g), (\mathrm{sk}, p, g))$*, where* $(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathcal{K}(1^k)$ *and* $(p, g) \leftarrow \mathcal{G}(1^k)$.

– $\mathcal{E}^*_{(\mathrm{pk},p,g)}(m) = \mathcal{E}_{\mathrm{pk}}(m)$.
– $\mathcal{D}^*_{(\mathrm{sk},p,g)}(c) = \begin{cases} \mathcal{D}_{\mathrm{sk}}(c) & \text{if } \mathcal{D}_{\mathrm{sk}}(c) \neq \perp \text{ and } 1 < \mathcal{D}_{\mathrm{sk}}(c) < p-1 \\ \perp & \text{otherwise} \end{cases}$.

Theorem 1. *Let $\mathcal{G}$ be a group parameter generator, and Π be a k-aware public-key encryption system. If Π is IND-CPA and $\Pi^{\mathcal{G}}$ is RR1, then TAP is $(\Pi, \mathcal{G})$-secure.*

We will prove the following logically equivalent statement: if $\Pi^{\mathcal{G}}$ is RR1, and TAP is $(\Pi, \mathcal{G})$-insecure, then Π is not IND-CPA.

Therefore, we now assume that we have in hand a knowledge extractor K for $\Pi^{\mathcal{G}}$ satisfying the properties of section 5, and an algorithm $\mathcal{M}^{f,H,\mathcal{N}}$ satisfying the properties of section 3, and will try to produce a pair of algorithms (A_1, A_2) that can win the guessing game of section 4. We will do this in two steps: (1) remove the node oracle; (2) win the guessing game.

6.1 Remove the Node Oracle

In this step, we take our algorithm $\mathcal{M}^{f,H,\mathcal{N}}$, which has access to the random oracles f and H and the node oracle $\mathcal{N}$, and produce an algorithm $\mathcal{M}_1^{f,H}$, which just has access to the random oracles.

$\mathcal{M}_1^{f,H}(\mathrm{pk}, p, g, c_0)$, then, is calculated as follows:

– Initialize Γ, Φ, and η to be empty lists.
– Set *out* $\leftarrow \mathcal{M}^{f',H',\mathcal{N}'}(\mathrm{pk}, p, g, c_0)$. Note that we have replaced each call to $f(m)$ by a call to the following subroutine $f'(m)$:

    append m to the list Φ
    return $f(m)$

  each call to $H(m)$ by a call to the following subroutine $H'(m)$:

    set $h \leftarrow H(m)$
    append (m, h) to η
    return h

  and each call to $\mathcal{N}(c)$ by a call to the following subroutine $\mathcal{N}'(c)$:

    set $k \leftarrow K(\eta, \{c_0\}, c, \mathrm{pk})$
    if $k = (\mathsf{plain}, \perp)$:
        return $\perp$
    else if $k = (\mathsf{plain}, v)$:
        pick a random exponent y
        set $s \leftarrow f(v^y)$
        return (g^y, s)

else:

pick a random exponent y
pick a random string s of length l_f
append g^y to the list Γ
return (g^y, s)

- Pick a random bit β.
- If $\beta = 0$, return *out*.
- If $\beta = 1$, and either Γ or Φ is empty, return $\perp$
- Otherwise, pick a random element γ of Γ and a random element ϕ of Φ, and return (γ, ϕ).

Lemma 1. *If $\mathcal{M}^{f,H,\mathcal{N}}(\mathrm{pk}, p, g, \mathcal{E}^H_{\mathrm{pk}}(g^x))$ outputs an x-power pair with non-negligible probability, then $\mathcal{M}_1^{f,H}(\mathrm{pk}, p, g, \mathcal{E}^H_{\mathrm{pk}}(g^x))$ outputs an x-power pair with non-negligible probability.*

Proof. The intuition behind the proof is that the input/output behaviours of f and f' are the same, as are those of H and H', so we only have to consider the difference between $\mathcal{N}$ and $\mathcal{N}'$. We can use K, the knowledge extractor for $\Pi^{\mathcal{G}}$, to give us a partial decryption oracle: if K reports that it knows the plaintext corresponding to the given ciphertext, $\mathcal{N}'$ can just use that value to perform the same operations as $\mathcal{N}$ would. On the other hand, if K reports that there's no way to know the plaintext, then $\mathcal{N}'$ can output a value which, by the properties of the random oracle, will be indistinguishable from those of $\mathcal{N}$, except in certain cases we consider separately.

We first note that $\mathcal{N}'$, and thus K, is called only polynomially often. Since each call to K only has a negligible probability of returning an erroneous result, we can conclude that, except with negligible probability, all of the calls to K return a correct result.

We assume, then, that indeed all of the calls to K return a correct result. Since K is a knowledge extractor for $\Pi^{\mathcal{G}}$, if K returns (plain, v), it must be the case that either $v = \perp$ or else $1 < v < p - 1$. In either case, $\mathcal{N}'$ performs the same operations as $\mathcal{N}$.

If K returns (guess), then $\mathcal{M}_1^{f,H}$ (in the role of K's adversary) cannot learn $\mathcal{D}^H_{\mathrm{sk}}(c)$ (and in particular $\mathcal{D}^H_{\mathrm{sk}}(c)$ cannot be $\perp$), except with negligible probability. In this case, $\mathcal{N}$ will return $(g^y, f(\mathcal{D}^H_{\mathrm{sk}}(c)^y))$, while $\mathcal{N}'$ will return (g^y, s), for a randomly chosen exponent y and a randomly chosen string s of length l_f. But $\mathcal{M}_1^{f,H}$ will not be able to compute $\mathcal{D}^H_{\mathrm{sk}}(c)^y$, so it will not be able to distinguish the two results. We also note that the probability that two different calls to $\mathcal{N}$ produce the same input to f is negligible: $\mathcal{N}$ only calls f on random powers of numbers d with $1 < d < p-1$. All such numbers have order either q or $2q = p-1$, and note that p was selected to be $\Omega(k)$ bits long. There are only polynomially many of these calls that $\mathcal{N}$ makes to f (since $\mathcal{M}^{f,H,\mathcal{N}}$ runs in polynomial time), so the probability that two of these calls have matching inputs is negligible.

Finally, if K returns (match, i), then i must be 1, since $\{\mathcal{E}^H_{\mathrm{pk}}(g^x)\}$ is a list of length 1, and so $\mathcal{D}^H_{\mathrm{sk}}(c) = g^x$. Again, $\mathcal{N}$ will return $(g^y, f(g^{xy}))$, and $\mathcal{N}'$ will return (g^y, s), for a randomly chosen exponent y and a randomly chosen string s of length l_f. What if $\mathcal{M}^{f,H,\mathcal{N}}$ makes a call to f that happens to match one of these inputs that $\mathcal{N}$ uses in a call to f? Suppose the probability of this event (which we will label C for "Collision") is Δ. Let θ_C be the conditional probability of $\mathcal{M}^{f,H,\mathcal{N}}$ succeeding (i.e. outputting an x-power pair), given that C has occurred. Similarly, let $\theta_{\overline{C}}$ be the conditional probability of $\mathcal{M}^{f,H,\mathcal{N}}$ succeeding, given that C has not occurred. Then the overall probability of $\mathcal{M}^{f,H,\mathcal{N}}$ succeeding is $\Delta \cdot \theta_C + (1-\Delta) \cdot \theta_{\overline{C}}$, which by assumption is non-negligible, so either $\Delta \cdot \theta_C$ or $(1-\Delta) \cdot \theta_{\overline{C}}$ (or both) must be non-negligible.

Note that, if C does not occur, then $\mathcal{M}_1^{f,H}$ will not be able to distinguish outputs of $\mathcal{N}'$ from outputs of $\mathcal{N}$, and so the probability of $\mathcal{M}^{f',H',\mathcal{N}'}$ outputting an x-power pair, given that C does not occur, is at least $\theta_{\overline{C}} - \epsilon$, for some negligible ϵ (which takes into account the negligible probabilities of error mentioned above).

On the other hand, if C does occur, then $\mathcal{M}_1^{f,H}$ will have made a call to f', passing an input g^{xy} (thus entering g^{xy} into the list Φ), where g^y is some value entered into the list Γ. Therefore, this (g^y, g^{xy}) pair is an x-power pair that appears in the set $\Gamma \times \Phi$.

So what is the overall probability of $\mathcal{M}_1^{f,H}$ succeeding? If C does not occur, and $\beta = 0$ (the combined probability of which is $\frac{1-\Delta}{2}$), then $\mathcal{M}_1^{f,H}$ will output *out*, which will be an x-power pair with probability at least $\theta_{\overline{C}} - \epsilon$. If C does occur, and $\beta = 1$ (the combined probability of which is $\frac{\Delta}{2}$), then $\mathcal{M}_1^{f,H}$ will output a random element of $\Gamma \times \Phi$, a set of polynomial size, of which at least one element is an x-power pair.

Therefore, the overall probability is at least $z = \frac{\Delta}{2} \cdot \frac{1}{|\Gamma \times \Phi|} + \frac{1-\Delta}{2} \cdot \left(\theta_{\overline{C}} - \epsilon\right)$. Now recall that either $\Delta \cdot \theta_C$ or $(1-\Delta) \cdot \theta_{\overline{C}}$ (or both) must be non-negligible. If $\Delta \cdot \theta_C$ is non-negligible, then $z \geq \frac{\Delta}{2} \cdot \frac{1}{|\Gamma \times \Phi|} = \frac{1}{2 \cdot |\Gamma \times \Phi|} \cdot \Delta \geq \frac{1}{2 \cdot |\Gamma \times \Phi|} \cdot (\Delta \cdot \theta_C)$, which is non-negligible. If $(1-\Delta) \cdot \theta_{\overline{C}}$ is non-negligible, then $z + \epsilon \geq z + \frac{1-\Delta}{2} \cdot \epsilon \geq \frac{1-\Delta}{2} \cdot \left(\theta_{\overline{C}} - \epsilon\right) + \frac{1-\Delta}{2} \cdot \epsilon = \frac{1}{2} \cdot (1-\Delta) \cdot \theta_{\overline{C}}$, which is non-negligible. In either case, z is non-negligible, as required. □

6.2 Win the Guessing Game

With $\mathcal{M}_1^{f,H}$ in hand, it is now straightforward to win the guessing game of section 4 against Π. Remember that Π is k-aware, so there is an polynomial-time algorithm κ that can extract the security parameter k from a public key pk generated by $\mathcal{K}(1^k)$.

Algorithm $A_1(\mathrm{pk})$**:**
set $k \leftarrow \kappa(\mathrm{pk})$

set $(p, g) \leftarrow \mathcal{G}(1^k)$
pick two (distinct) random exponents x_0, x_1
return $(m_0, m_1, \sigma) \leftarrow (g^{x_0}, g^{x_1}, (\mathrm{pk}, p, g, x_0, x_1))$

Algorithm $A_2(m_0, m_1, \sigma, y)$**:**
set $(\mathrm{pk}, p, g, x_0, x_1) \leftarrow \sigma$
set $out \leftarrow \mathcal{M}_1^{f,H}(\mathrm{pk}, p, g, y)$
if $out = (\alpha, \alpha^{x_0})$ for some $1 < \alpha < p - 1$:
    return 0
else if $out = (\alpha, \alpha^{x_1})$ for some $1 < \alpha < p - 1$:
    return 1
else:
    return a random element of $\{0, 1\}$

Why does this work? The tester (in the nomenclature of section 4) will pick a random bit b, and pass $\mathcal{E}_{\mathrm{pk}}^H(m_b) = \mathcal{E}_{\mathrm{pk}}^H(g^{x_b})$ to A_2 as y. A_2 will then calculate $out = \mathcal{M}_1^{f,H}(\mathrm{pk}, p, g, \mathcal{E}_{\mathrm{pk}}^H(g^{x_b}))$, which, by the above, will be a x_b-power pair with non-negligible probability δ. Also, since x_0 and x_1 were picked randomly, and $\mathcal{M}_1^{f,H}$ never learns any value that depends on x_{1-b}, the probability that *out* is an x_{1-b}-power pair must be some negligible value ϵ. So the probability of A_2 outputting b is then $\delta + \frac{1-\delta-\epsilon}{2} = \frac{1}{2} + \frac{\delta-\epsilon}{2}$, and $\mathsf{Adv}_{(A_1,A_2),\Pi}^{\mathrm{ind-cpa}} = \delta - \epsilon$, which is non-negligible, so Π is *not* IND-CPA.

Therefore, we have that if $\Pi^{\mathcal{G}}$ is RR1, and TAP is $(\Pi, \mathcal{G})$-insecure, then Π is not IND-CPA, which completes the proof of Theorem 1. □

7 The Concrete Protocol

In this final section, we examine the actual encryption mechanism used by Tor, and show that it indeed satisfies the preconditions of Theorem 1, under reasonable assumptions.

First, we need to work around a slight technicality: so far, all of our analyses have been parameterized by the security parameter k. Unfortunately, the Tor specification [10] is not so parameterized: it specifies a single Diffie-Hellman group, for example. The algorithms we outline here, therefore, are generalizations of the actual Tor algorithms, and reduce to the actual algorithms for a specific value of k.

Let $T_{RSA}(l_N)$ be a lower bound on the expected amount of work an adversary must do to break RSA with an l_N-bit modulus. We of course assume this bound

is superpolynomial in l_N.[5] Then select values for parameters $(l_N, l_p, l_x, l_f, l_H, l_s)$, based on a security parameter k, as follows:

- l_N will be the bitlength of an RSA modulus. Select a value divisible by 8, such that $\lfloor \log_2 T_{RSA}(l_N) \rfloor = k$.
- l_p will be the bitlength of a Diffie-Hellman modulus. Select $l_p = l_N$.
- l_x will be the bitlength of random exponents. Select a value divisible by 8, such that l_x is $\Omega(k)$.
- l_f and l_H will be the bitlengths of the outputs of random oracles. Select values divisible by 8, such that l_f and l_H are each $\Theta(k)$.
- l_s will be the bitlength of a symmetric key. Select a value divisible by 8, such that l_s is $\Theta(k)$.

It must be the case that $l_N - 2l_H - 16 - l_s$ is positive, and $\Omega(k)$. Define r to be $\frac{1}{8}(l_N - 2l_H - 16 - l_s)$.

In the specified protocol, $k = 85$ and $(l_N, l_p, l_x, l_f, l_H, l_s) = (1024, 1024, 320, 160, 160, 128)$. The random oracles f and H are instantiated by appropriately chosen hash functions with output lengths l_f and l_H bits, respectively. Let S^* be a family of pseudorandom functions (such as a block cipher) with keylength l_s. The specified protocol uses hash functions based on SHA-1 [16], and uses AES-128 [17] as the symmetric encryption function.[6]

For $m \in \mathbb{N}$, define the pair (m^L, m^R) as follows:

- Express the integer m as a sequence mo of octets, most significant first. This sequence should be of minimum length; i.e. no leading `0x00`s.
- Let m^L be the first r octets of mo, and let m^R be the remainder of mo.

The public-key encryption system used in Tor is then the following $\Pi_{\mathsf{TAP}} = (\mathcal{K}, \mathcal{E}^H, \mathcal{D}^H)$:

$\mathcal{K}(1^k)$ outputs a randomly generated RSA keypair $(\mathrm{pk}, \mathrm{sk}) = ((N, e), (N, d))$ where the bitlength of N is l_N (which depends on k, as above), and $e = 65537$.

$\mathcal{E}^H_{\mathrm{pk}}(m)$ is as follows:

- Pick a random key s for S^*, of length l_s.
- Let C_1 be the RSA-OAEP encryption (using the hash function H internally, and the key pk) of the concatenation of s and m^L.
- Let C_2 be the encryption (using the cryptosystem S^* in CTR mode with key s and initial counter 0) of m^R.

[5] For concreteness, we use $T_{RSA}(l_N) = \exp\left(\xi\ (\ln 2^{l_N})^{1/3} (\ln \ln 2^{l_N})^{2/3}\right)$, where $\xi = \frac{1}{3}\left(92 + 26\sqrt{13}\right)^{1/3} \approx 1.902$. This is the asymptotic expected running time for factoring an l_N-bit integer using the generalized number field sieve [6].

[6] AES-128 is indeed an appropriate choice for S^*, as long as an attacker cannot distinguish AES-128 from a family of random functions.

– Output (C_1, C_2).

$\mathcal{D}^H_{\mathrm{sk}}((C_1, C_2))$ is as follows:

– Decrypt C_1 using RSA-OAEP (with the hash function H and the key sk). Let s be the first l_s bits of the result, and mo_1 be the remainder of the result.
– Decrypt C_2 using the cryptosystem S^* in CTR mode with key s and initial counter 0, yielding mo_2.
– Concatenate mo_1 and mo_2, and turn the result into an MSB-first integer m.
– If any step failed, return $\perp$. Otherwise, return m.

The group parameter generator $\mathcal{G}_{\mathsf{TAP}}$ for Tor returns a deterministic (p, g) for any input 1^k, with p of bitlength l_p, and such that $1 - p \cdot 2^{-l_p}$ is a negligible function of k. This last condition means that a random integer of the same length as p has only a negligible probability of being greater than p.

Now that we have specified TAP for generic k, we must simply check that Π_{TAP} and $\mathcal{G}_{\mathsf{TAP}}$ satisfy the preconditions of Theorem 1, namely:

1. Π_{TAP} is k-aware.
2. Π_{TAP} is IND-CPA.
3. $\Pi_{\mathsf{TAP}}^{\mathcal{G}_{\mathsf{TAP}}}$ is RR1.

7.1 Π_{TAP} is k-aware

We need to produce an algorithm κ which outputs k when given a public key output by $\mathcal{K}(1^k)$.

This is easy: given a public key (N, e) output by $\mathcal{K}(1^k)$, let l_N be the bitlength of N. Then output $\lfloor \log_2 T_{RSA}(l_N) \rfloor$, which will equal k, by the choice of l_N.

7.2 Π_{TAP} is IND-CPA

We first note that, for any one-way trapdoor permutation g, g-OAEP is IND-CPA [4], and that for any pseudorandom function F, F-CTR is IND-CPA[7] [8].

We prove, more generally, that the hybrid construction of Π_{TAP} is IND-CPA, for any choice of underlying public-key encryption system (R) and symmetric-key encryption system (S), so long as they are each IND-CPA themselves.[8]

[7] We have not formally defined the notion of IND-CPA for symmetric encryption, but the definition is analogous to that in section 4; the only changes are that $\mathcal{K}$ returns a single key instead of a keypair, and 1^k (and not pk) is the input to A_1.

[8] Our proof is similar to that of Theorem 5 of [7], though that result pertained to IND-CCA systems, and did not need to deal with part of the plaintext messages being encrypted under the underlying public-key system.

Let (A_1, A_2) be any polynomial-time adversary in the IND-CPA game of section 4 against Π_{TAP}. Let $[s, \rho]$ denote the concatenation of the l_s-bit value s and the r-octet value ρ. Then define an adversary $A' = (A'_1, A'_2)$ against S and a pair of adversaries $A''_\beta = (A''_{1,\beta}, A''_{2,\beta})$ (for $\beta \in \{0, 1\}$) against R as follows:

Algorithm $A'_1(1^k)$**:**
$(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathcal{K}(1^k)$
$(m_0, m_1, \sigma) \leftarrow A_1(\mathrm{pk})$
return $(m_0^R, m_1^R, (\mathrm{pk}, m_0, m_1, \sigma))$

Algorithm $A'_2(m_0^R, m_1^R, (\mathrm{pk}, m_0, m_1, \sigma), y)$**:**
pick a random l_s-bit key s and a random r-octet string ρ
return $A_2(m_0, m_1, \sigma, (R_{\mathrm{pk}}([s, \rho]), y))$

Algorithm $A''_{1,\beta}(\mathrm{pk})$**:**
$(m_0, m_1, \sigma) \leftarrow A_1(\mathrm{pk})$
pick two random l_s-bit keys s_0, s_1 and a random r-octet string ρ
return $([s_0, m_\beta^L], [s_1, \rho], (m_0, m_1, \sigma))$

Algorithm $A''_{2,\beta}([s_0, m_\beta^L], [s_1, \rho], (m_0, m_1, \sigma), y)$**:**
return $A_2(m_0, m_1, \sigma, (y, S_{s_0}(m_\beta^R)))$

For $\beta \in \{0, 1\}$, define μ_β to be the conditional probability that $A_2(m_0, m_1, \sigma, y) = 0$, given that (m_0, m_1, σ) is an output of $A_1(\mathrm{pk})$, and that $y = (R_{\mathrm{pk}}([s, m_\beta^L]), S_s(m_\beta^R))$ for a randomly chosen l_s-bit key s. That is, μ_β is the probability that A_2 outputs 0 as its guess for the tester's value b, when the correct answer was β.

Also for $\beta \in \{0, 1\}$, define ν_β to be the conditional probability that $A_2(m_0, m_1, \sigma, y) = 0$, given that (m_0, m_1, σ) is an output of $A_1(\mathrm{pk})$, and that $y = (R_{\mathrm{pk}}([s_1, \rho]), S_{s_0}(m_\beta^R))$ for randomly chosen l_s-bit keys s_0 and s_1 and a randomly chosen r-octet string ρ. That is, $|\nu_\beta - \mu_\beta|$ is the probability that A_2 can distinguish between a correctly formed value of y, encrypting m_β (namely, $(R_{\mathrm{pk}}([s, m_\beta^L]), S_s(m_\beta^R))$), and one in which the wrong symmetric key (and the wrong m_β^L) is encrypted with the public-key system (namely, $(R_{\mathrm{pk}}([s_1, \rho]), S_{s_0}(m_\beta^R))$).

We now note that $\mathsf{Adv}^{\mathrm{ind-cpa}}_{A,\Pi_{\mathsf{TAP}}} = \left|2\left(\frac{1}{2}\mu_0 + \frac{1}{2}(1 - \mu_1)\right) - 1\right| = |\mu_0 - \mu_1|$. Similarly, $\mathsf{Adv}^{\mathrm{ind-cpa}}_{A',S} = |\nu_0 - \nu_1|$, and $\mathsf{Adv}^{\mathrm{ind-cpa}}_{A''_\beta,R} = |\mu_\beta - \nu_\beta|$.

Therefore, $\mathsf{Adv}^{\mathrm{ind-cpa}}_{A,\Pi_{\mathsf{TAP}}} \leq \mathsf{Adv}^{\mathrm{ind-cpa}}_{A''_0,R} + \mathsf{Adv}^{\mathrm{ind-cpa}}_{A',S} + \mathsf{Adv}^{\mathrm{ind-cpa}}_{A''_1,R}$. So if Π_{TAP} is not IND-CPA, then for some adversary A, $\mathsf{Adv}^{\mathrm{ind-cpa}}_{A,\Pi_{\mathsf{TAP}}}$ is non-negligible, which means at least one of $\mathsf{Adv}^{\mathrm{ind-cpa}}_{A',S}$, $\mathsf{Adv}^{\mathrm{ind-cpa}}_{A''_0,R}$, and $\mathsf{Adv}^{\mathrm{ind-cpa}}_{A''_1,R}$ must be non-negligible, which means at least one of R and S is not IND-CPA, as required.

So under the usual assumption that RSA is a one-way trapdoor permutation, Π_{TAP} is also IND-CPA.

7.3 $\Pi_{\mathsf{TAP}}^{\mathcal{G}_{\mathsf{TAP}}}$ is RR1

We must produce a knowledge extractor K for $\Pi_{\mathsf{TAP}}^{\mathcal{G}_{\mathsf{TAP}}}$ with the properties of section 5.

From [12], we know that, under the RSA assumption, there is a decryption simulator DS for RSA-OAEP (but *not* f-OAEP in general!) such that $DS(\eta, c^*, c, \mathrm{pk}) = \mathcal{D}_{\mathrm{sk}}^H(c)$, for any $(\mathrm{pk}, \mathrm{sk})$ output by $\mathcal{K}(1^k)$, any distinct ciphertexts c, c^*, and the list of oracle queries and responses η used to generate c (but *not* c^*), except with negligible probability.

Given this, the construction of K is straightforward:

$K(\eta, \{(C_1^*, C_2^*)\}, (C_1, C_2), (\mathrm{pk}, p, g))$:
    determine l_p, l_s, and r based on $k \leftarrow \kappa(\mathrm{pk})$
    let r_2 be the length (in octets) of C_2
    if $C_1 = C_1^*$:
(1) if $8(r + r_2) - l_s > l_p$: return $(\mathsf{plain}, \perp)$
(2) else if $C_2 = C_2^*$: return $(\mathsf{match}, 1)$
(3) else: return (guess)
    else:
        let $m_1 \leftarrow DS(\eta, C_1^*, C_1, \mathrm{pk})$
        if $m_1 = \perp$: return $(\mathsf{plain}, \perp)$
        let s be the first l_s bits of m_1, and let mo_1 be the remainder of m_1
        decrypt C_2 using S^*-CTR$_s$ and initial counter 0, yielding mo_2
        if $mo_2 = \perp$: return $(\mathsf{plain}, \perp)$
        concatenate mo_1 and mo_2, yielding mo
        turn mo into an MSB-first integer m
        if $1 < m < p - 1$: return (plain, m)
        else: return $(\mathsf{plain}, \perp)$

Remember that (C_1^*, C_2^*) can be assumed to be a valid encryption of the plaintext message g^x for some random exponent x, so if $C_1 = C_1^*$, then (C_1, C_2) will decrypt (under Π_{TAP}) to a value v whose first $8r - l_s$ bits will be the same as those of g^x. Also, still assuming $C_1 = C_1^*$:

- If the length of v is larger than the length of p, then certainly $v > p$, and $\Pi_{\mathsf{TAP}}^{\mathcal{G}_{\mathsf{TAP}}}$ would return $\perp$, so we return $(\mathsf{plain}, \perp)$ in line (1).
- If $C_2 = C_2^*$, then of course $\mathcal{D}_{\mathrm{sk}}^H((C_1, C_2)) = \mathcal{D}_{\mathrm{sk}}^H((C_1^*, C_2^*))$, so we return $(\mathsf{match}, 1)$ in line (2).
- Otherwise, the length of v is at least $8r - l_s$, and at most the length of p, so $1 < v < p - 1$ except with negligible probability, by our choice of p, so

$v = \mathcal{D}^H_{\mathrm{sk}}((C_1, C_2))$. Further, $v \neq \mathcal{D}^H_{\mathrm{sk}}((C_1^*, C_2^*))$, since S^*-CTR with an initial counter of 0 is a deterministic encryption function. Finally, no algorithm can predict v, since its first $8r - l_s$ bits are the same as those of the randomly chosen g^x, so we return (guess) in line (3).

If C_1 does not equal C_1^*, then we can use the decryption simulator for RSA-OAEP to decrypt it successfully, except with negligible probability, and then we just perform the same actions as the real $\Pi^{\mathcal{G}_{\mathsf{TAP}}}_{\mathsf{TAP}}$ would.

Therefore, this K satisfies the properties of Definition 5 for $\Pi^{\mathcal{G}_{\mathsf{TAP}}}_{\mathsf{TAP}}$, so $\Pi^{\mathcal{G}_{\mathsf{TAP}}}_{\mathsf{TAP}}$ is RR1, as required.

8 Conclusion

Under the assumptions that RSA is one way, and that an appropriately strong block cipher is used, we have shown that the Tor Authentication Protocol is secure in the random oracle model; that is, without exploiting particular structure of the hash functions, a man-in-the-middle has only a negligible chance of being able to read messages that Alice thinks she's sending to Bob.

It should be noted, however, that the proof is sensitive to specific properties of TAP, and any modifications to the protocol should take care not to destroy these properties. For example, if Bob were to check that the order of the received message m were equal to exactly q, as opposed to merely checking that $1 < m < p - 1$, $\Pi^{\mathcal{G}_{\mathsf{TAP}}}_{\mathsf{TAP}}$ would *not* be RR1. On the other hand, replacing Π_{TAP} with a stronger system, such as one that is plaintext aware, would make TAP more robust to other modifications.

Acknowledgements. We would like to thank Stefan Brands for originally suggesting the problem, Jan Camenisch for his helpful initial discussion, Nikita Borisov and Dennis Kügler for clarifying the presentation of the paper, and the anonymous referees.

References

1. Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. In *Advances in Cryptology—CRYPTO '98, Lecture Notes in Computer Science 1462*, pages 26–45. Springer-Verlag, August 1998.
2. Mihir Bellare and Adriana Palacio. Towards Plaintext-Aware Public-Key Encryption without Random Oracles. In *Advances in Cryptology—Asiacrypt 2004, Lecture Notes in Computer Science 3329*, pages 48–62. Springer-Verlag, 2004.

3. Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
4. Mihir Bellare and Phillip Rogaway. Optimal Asymmetric Encryption—How to Encrypt with RSA. In *Advances in Cryptology—Eurocrypt '94, Lecture Notes in Computer Science 950*. Springer-Verlag, 1994.
5. Daniel Bleichenbacher. Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS#1. In *Advances in Cryptology—CRYPTO '98, Lecture Notes in Computer Science 1462*, pages 1–12. Springer-Verlag, August 1998.
6. Don Coppersmith. Modifications to the Number Field Sieve. *Journal of Cryptology*, 6(3):169–180, 1993.
7. Ronald Cramer and Victor Shoup. Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.
8. Anand Desai and Sara Miner. Concrete Security Characterizations of PRFs and PRPs: Reductions and Applications. In *Advances in Cryptology—Asiacrypt 2000, Lecture Notes in Computer Science 1976*, pages 503–516. Springer-Verlag, 2000.
9. Roger Dingledine. Tor security advisory: DH handshake flaw. http://archives.seul.org/or/announce/Aug-2005/msg00002.html, August 2005.
10. Roger Dingledine and Nick Mathewson. Tor Protocol Specification, version 1.112. http://tor.eff.org/cvs/tor/doc/tor-spec.txt, January 2006.
11. Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.
12. Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. RSA-OAEP is Secure under the RSA Assumption. In *Advances in Cryptology—CRYPTO 2001, Lecture Notes in Computer Science 2139*, pages 260–274. Springer-Verlag, August 2001.
13. Chris Hall, Ian Goldberg, and Bruce Schneier. Reaction Attacks Against Several Public-Key Cryptosystems. In *International Conference on Information and Communication Security 1999*, November 1999.
14. Jeffrey Hoffstein and Joseph H. Silverman. Reaction Attacks Against the NTRU Public Key Cryptosystem. NTRU Cryptosystems Technical Report #015, Version 2, http://www.ntru.com/cryptolab/pdf/NTRUTech015.pdf, June 2000.
15. Paul Syverson. Personal communication.
16. U.S. Department of Commerce, N.I.S.T. Secure Hash Algorithm. FIPS 180-1, 1995.
17. U.S. Department of Commerce, N.I.S.T. Advanced Encryption Standard (AES). FIPS 197, 2001.