Using Sphinx to Improve Onion Routing Circuit Construction (short paper)*

Aniket Kate and Ian Goldberg

David R. Cheriton School of Computer Science
University of Waterloo, ON, Canada
{akate,iang}@cs.uwaterloo.ca

Abstract. This paper presents compact message formats for onion routing circuit construction using the Sphinx methodology developed for mixes. We significantly compress the circuit construction messages for three onion routing protocols that have emerged as enhancements to the Tor anonymizing network; namely, Tor with predistributed Diffie-Hellman values, pairing-based onion routing, and certificateless onion routing. Our new circuit constructions are also secure in the universal composability framework, a property that was missing from the original constructions. Further, we compare the performance of our schemes with their older counterparts as well as with each other.

1 Introduction

Goldschlag, Reed and Syverson [13] proposed *onion routing* to achieve low-latency anonymous communication on public networks, which motivated the original Onion Routing project [18] and many other anonymous communication constructions [7, 9, 11]. Among these, with its hundreds of thousands of users, the second generation onion routing project—Tor [20]—has turned out to be a huge success. However, with its latency times of a few seconds, users find Tor to be very slow for their usual communication over the Internet, and employ it only in situations where their anonymity is indispensable to them. Efficiency is essential for widespread use of anonymity networks; therefore, defining an efficient practical onion routing protocol forms the motivation of this work.

An onion routing (OR) network consists of a set of *onion routers* (OR nodes) that relay traffic, a large set of users and a *directory server* that provides routing information of the OR nodes to the users. A user constructs a *circuit* choosing a small ordered subset of OR nodes, where the chosen nodes route the user's traffic over the path formed. The key property is that it is difficult for any OR node in a circuit to determine the circuit nodes other than its predecessor and successor. Further, the task must also be difficult for a powerful but not global observer. The user achieves this by sending the first OR node an *onion*—a message wrapped in multiple layers of encryption (one layer per selected node). A user includes the identifier of the next node and a random symmetric session key in each onion layer, and uses nodes' public keys to encrypt their respective layers. A node decrypts a received onion using its private key, forwards the remaining onion to

* See [14] for the full version of this paper.

the next node, and uses the random symmetric session key for the rest of the session. However, this *single-pass* circuit construction is not *forward secret*; if an adversary corrupts a node and obtains its private key, then the adversary can decrypt all of its past communication. The adversary could then successively compromise all the nodes in a circuit to break the anonymity of a user's past communications. Although changing the the public/private key pairs for all OR nodes after a predefined interval (*forward secrecy phase*) is a possible solution, it is not scalable. Every system user now has to download a new set of public keys for all the nodes at the start of every forward secrecy phase.

Observing the above issue with forward secrecy, Dingledine, Mathewson and Syverson [9] used an interactive and incremental *telescoping* approach while designing Tor. In the Tor authentication protocol (TAP), which is used to negotiate the session keys in this *multi-pass* circuit construction, a node's public key is only used to initiate the construction and its compromise does not void the security of the session keys once the randomness used in the protocol is erased. Øverlier and Syverson [17] improved the efficiency of Tor using a half-certified Diffie-Hellman (DH) key agreement [16, §12.6].

However, in Tor, $\Theta(\nu^2)$ messages are required to create a circuit of length ν, as compared to $\Theta(\nu)$ required in a single-pass circuit construction. To solve the scalability issue in single-pass circuit constructions, Kate, Zaverucha and Goldberg [15] suggested the use of an identity-based setting and defined a pairing-based onion routing protocol (PB-OR). Catalano, Fiore and Gennaro [6] suggested the use of a certificateless setting instead and defined two certificateless onion routing protocols (CL-OR and 2-CL-OR). Øverlier and Syverson [17] have also suggested a single-pass circuit construction that provides forward secrecy eventually. However, an extensive comparison between all these schemes is not available yet. In terms of security, none of these practical protocols achieve security in the universal composability (UC) framework [5]. Camenisch and Lysyanskaya [4] presented a framework for UC-secure OR circuit construction, but their protocol is not practical enough for realistic use.

Contributions. In this paper, we present a practical generic onion routing circuit construction protocol that achieves security in the UC model. We apply our protocol to Tor-preDH, PB-OR, CL-OR and 2-CL-OR to define their UC-secure versions. Importantly, the circuit construction messages for these new protocols are significantly smaller than those in the original protocol and there is no addition to a user's computational cost. We achieve this using *Sphinx*, an efficient message format for mix networks, defined by Danezis and Goldberg [8].

2 Preliminaries

The Sphinx Message Format. Mix message formats have been a point of interest in research on mix networks (see references in [8]). Recently, Danezis and Goldberg [8] proposed Sphinx as the most compact, efficient and UC-secure cryptographic mix message format.

In Sphinx, an adversary is computationally bounded by a security parameter κ. For a prime q of size 2κ bits, let $\mathbb{G}$ be a cyclic group of order q that satisfies

the *decisional Diffie-Hellman* (DDH) assumption. Sphinx makes the circuit construction message size independent of the length of the circuit, ν; we denote the maximum length of a circuit as r. Node identifiers are κ-bit strings. Each node has a public/private key pair. Further, Sphinx assumes a message authentication code (MAC) μ, a pseudo-random generator (PRG) ρ and corresponding random oracle hash functions $h_\mu, h_\rho : \mathbb{G}^* \rightarrow \{0,1\}^\kappa$, where $\mathbb{G}^*$ is the set of non-identity elements of $\mathbb{G}$. It also needs a random oracle hash function $h_b : \mathbb{G}^* \times \mathbb{G}^* \rightarrow \mathbb{Z}_q^*$.

Cryptographically, the most elegant feature of the Sphinx message format is its session key derivation technique based on a repeatedly modified random element of a cyclic prime order group. We call this technique Sphinx's *blinding logic*. The mentioned random element (α) and its repeated modified forms are called *pseudonyms* since each of these random elements is a temporary public key whose private key is held by the user. In the Sphinx blinding logic, each mix node uses a pseudonym supplied by its predecessor and its own private key to compute the session key with the user. To improve the unlinkability, as in Tor, a pseudonym must not remain the same across the circuit. In the mix network and onion routing literature, this is done by including separate random pseudonyms in a construction message for each node in the circuit. In Sphinx's blinding logic, this is achieved using a single repeatedly changing pseudonym. At every node, a blinding factor is extracted from the current pseudonym and the newly computed session key. The current pseudonym is then exponentiated with the blinding factor to generate the next pseudonym. In other words, $\alpha_{i+1} = \alpha_i^{h_b(\alpha_i, s_i)}$. The session key is computed by node n_i as $s_i = \alpha_i^{x_i}$, where x_i is the node's private key, and by the user as explained in §3.

To send an anonymous message, a sender first chooses her mix nodes and obtains their public keys. She then computes α_i and s_i and wraps the message in multiple layers of encryption using the PRG ρ to generate ciphertext values β_i. To check the integrity of the message header, she calculates and includes a MAC γ_i at each mixing stage. Upon receiving a message header $(\alpha_i, \beta_i, \gamma_i)$, each mix node n_i extracts session keys using its private key x_i and the pseudonym α_i received from the predecessor. It uses those to verify the MAC γ_i and to decrypt a layer of encryption of β_i. It also extracts the routing information, computes the pseudonym α_{i+1} for the next node and forwards the message to n_{i+1}.

Tor Circuit Construction and Recent Enhancements. In Tor circuit construction [9], a user performs a DH key agreement with each successive node in her circuit over a secure tunnel formed using the already-agreed session keys. This ensures the forward secrecy of the communication immediately after these session keys are deleted. In TAP, a user extends a circuit to node n_i by generating a random $x_i \in_R \mathbb{Z}_q^*$ and sending a DH value (that is, a pseudonym) g^{x_i} encrypted using the (RSA) public key of node n_i. Node n_i decrypts the message and responds by sending g^{y_i}, where $y_i \in_R \mathbb{Z}_q^*$, and a hash of $g^{x_i y_i}$. It is important that the user herself generates and encrypts the DH value g^{x_i}; if an intermediate adversary OR node (n_j for $0 < j < i$) derives g^{x_i}, it can launch a *man-in-the-middle* attack. In Sphinx's blinding logic, node n_{i-1} uses the received pseudonym $g^{x_{i-1}}$ to generate and send pseudonym g^{x_i} to node n_i unencrypted. Therefore, it is not possible to directly apply the compact Sphinx message format to Tor.

Øverlier and Syverson [17], Kate *et al.* [15], and Catalano *et al.* [6] suggested improvements to the Tor circuit construction. These schemes use a *one-way anonymous* key agreement [15] strategy in the public-key cryptography (PKC), identity-based cryptography (IBC) and certificateless cryptography (CLC) settings respectively. Here, a user chooses a random element of $\mathbb{Z}_q^*$ per circuit node and computes an associated pseudonym. A session key is computed using the node's public key and the random element at the user end, and using the pseudonym received and the node's private key at the node's end; the precise session-key computation and the cryptographic assumption vary with the OR circuit construction protocol. Most importantly, unlike Tor, the user does not encrypt the pseudonyms in these schemes, which is a direct result of the inclusion of the private key of an OR node in the session key generation. Therefore, it is possible to incorporate Sphinx's blinding logic into these schemes.

3 Using Sphinx in OR Circuit Construction

In this section, we first present the generic design of OR circuit construction using the Sphinx methodology. We then implement the generic Sphinx format into three onion routing circuit constructions. Our design goals and threat model are the same as those of Tor. Refer to the full version of this paper [14] for details.

Generic Design. In Sphinx, a pseudonym α_{i+1} for node n_{i+1} is generated using the pseudonym α_i and the session key s_i generated at node n_i. As discussed above, we can use the Sphinx methodology in an OR circuit construction protocol where a node can create or observe a pseudonym for the next node in the circuit.

To create an OR circuit construction message, we use Sphinx's mix header creation algorithm ([8, §3.2]) with a generalization of the session key generation. The original Sphinx message format is based on the half-certified DH key agreement [16, §12.6], where a session key s_i is generated as $s_i = y_i^{xb_0b_1\cdots b_{i-1}}$ at the user's end and as $s_i = \alpha_i^{x_i}$ at node n_i, where (y_i, x_i) is the public/private key pair for n_i, α_i is a pseudonym for node n_i, x is the session-specific randomness and $b_0, \ldots, b_{i-1}$ are the blinding factors, $b_i = h_b(\alpha_i, s_i)$. The different OR circuit construction protocols use different session key generation methods, so we generalize this session key generation step. At the user end, we set $s_i = f_U(y_i, xb_0b_1 \cdots b_{i-1})$, and at node n_i, $s_i = f_N(x_i, y_i, \alpha_i)$. The other technical details of Sphinx remain exactly the same. Refer to [8, §3.2] for circuit construction message creation and to [8, §3.6] for message processing at a node.

Note that, although Sphinx is defined for single-pass constructions, its blinding logic is also useful in multi-pass constructions, where it can avoid the transfer of pseudonyms in circuit extension messages. However, we concentrate only on single-pass constructions as the applicability of Sphinx is more evident there.

Security Analysis. Camenisch and Lysyanskaya [4] design a UC-secure framework for onion routing. They define onion-correctness, onion-integrity and onion-security properties for an OR scheme and prove Theorem 1.

Theorem 1 (Theorem 1 [4]). *An onion routing scheme satisfying onion-correctness, integrity and security, when combined with secure point-to-point secure channels, yields a UC-secure OR scheme.*

Danezis and Goldberg [8] separate a wrap-resistance property from onion-security to simplify the onion-security definition and prove the resulting four security properties of the Sphinx message format using random oracles. We use their security discussion to define the security requirements for our generic OR circuit design. Refer to [14] for details.

We next apply the above generic design to three OR circuit constructions.

Tor with predistributed DH values (Tor-preDH). The half-certified DH key agreement scheme [16, §12.6] is a one-pass protocol with unilateral key authentication of the receiver to the sender, assuming that the sender has an authentic copy of the receiver's public key. Øverlier and Syverson [17] define an enhancement to the Tor circuit construction using this technique.

Let $x_i \in \mathbb{Z}_q^*$ be the private key for node n_i and let $y_i = g^{x_i}$ be its public key, where $g \in \mathbb{G}$ is a chosen generator. In the half-certified DH key agreement scheme, a user generates a random $r_i \in_R \mathbb{Z}_q^*$ and sends a pseudonym $\alpha_i = g^{r_i}$ to node n_i over the already formed circuit (tunnel), if any. The user generates the session key as $s_i = y_i^{r_i}$ and node n_i generates $s_i = \alpha_i^{x_i}$. Øverlier and Syverson used this to present a single-pass protocol (their second protocol).

Using the generic Sphinx design, we can not only make their eventual forward secret protocol more efficient but also prove its security in the UC model. Here, except for the entry node, the user is not required to send α_i to node n_i in the circuit. Node n_{i-1} generates the pseudonym $\alpha_i = \alpha_{i-1}^{h_b(\alpha_{i-1}, s_{i-1})}$. All other computation remains the same as the half-certified DH key agreement and the message format remains the same as that of Sphinx.

Pairing-based Onion Routing. Kate *et al.* [15] observe that the public-key management issue while achieving forward secrecy in single-pass onion routing circuit constructions can be solved using IBC. They develop an anonymous key agreement protocol modifying Sakai-Ohgishi-Kasahara key agreement [19] in the Boneh-Franklin identity-based encryption (BF-IBE) setup [3] and use that to define an OR circuit construction called *pairing-based onion routing* (PB-OR).

We choose three cyclic groups $\mathbb{G}$, $\hat{\mathbb{G}}$, and $\mathbb{G}_T$ (all of which we shall write multiplicatively) of prime order q and a *bilinear pairing* $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$. We refer the readers to [12] for a detailed discussion of pairings. In the BF-IBE setup, given ($e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$, $g \in \mathbb{G}$, $\hat{g} \in \hat{\mathbb{G}}$), a (possibly distributed) private-key generator (PKG) generates a master key $s \in \mathbb{Z}_q^*$ and an associated public key $y = g^s \in \mathbb{G}^*$, and derives private keys d_i for nodes using their well-known identities and s. A node with identity $\mathtt{ID}_i$ receives the private key $d_i = (h_{\mathtt{ID}}(\mathtt{ID}_i))^s \in \hat{\mathbb{G}}^*$, where $h_{\mathtt{ID}} : \{0,1\}^* \to \hat{\mathbb{G}}^*$ is a cryptographic hash function.

In PB-OR, a user generates a random $r_i \in_R \mathbb{Z}_q^*$ and sends a pseudonym $\alpha_i = g^{r_i}$ to node n_i over the already-formed circuit (if any). The session key s_i is generated at the user end as $s_i = e(y, h_{\mathtt{ID}}(\mathtt{ID}_i))^{r_i}$ and at the node n_i as $s_i = e(\alpha_i, d_i)$. Using our generic design, α_i can be generated as $\alpha_i = \alpha_{i-1}^{h_b(\alpha_{i-1}, s_{i-1})}$, while the computation of s_i remains the same as that of the original PB-OR, except here $r_i = x b_0 b_1 \cdots b_{i-1}$ for an $x \in_R \mathbb{Z}_q^*$ chosen by the user.

Certificateless Onion Routing. Catalano *et al.* [6] recently introduced the concept of certificateless onion routing and presented two protocols (CL-OR

and 2-CL-OR) for it. Their motivation is to avoid pairings and to eliminate the interactions between a PKG (or key generation centre—KGC) and nodes in PB-OR using CLC introduced by Al-Riyami and Paterson [1].

In certificateless onion routing, the KGC chooses a random generator $g \in_R \mathbb{G}$, two hash functions $h_{CL} : \{0,1\}^* \rightarrow \mathbb{Z}_q$ and $h_\pi : \mathbb{G} \times \mathbb{G} \rightarrow \{0,1\}^\kappa$, and a master key $s \in_R \mathbb{Z}_q$. It then computes $y = g^s$ and publishes $(\mathbb{G}, g, y, h_{CL}, h_\pi)$ as the public key. When a node n_i with identity $\mathtt{ID}_i$ asks for its partial private key, the KGC first generates a random $k_i \in_R \mathbb{Z}_q$, computes $\omega_i = g^{k_i}$ and $z_i = k_i + h_{CL}(\mathtt{ID}_i, \omega_i)s$ and returns $d_i = (\omega_i, z_i)$ to node n_i. Each node also generates a random $t_i \in_R \mathbb{Z}_q$ and computes $u_i = g^{t_i}$. The public key for a node n_i with identity $\mathtt{ID}_i$ is (ω_i, u_i) and its private key is (z_i, t_i). In CL-OR, a user generates a random $r_i \in_R \mathbb{Z}_q^*$ and sends the corresponding pseudonym $\alpha_i = g^{r_i}$ to node n_i. The user generates the session key $s_i = (z_{i1}, z_{i2})$ such that $z_{i1} = (\omega_i y^{h_{CL}(\mathtt{ID}_i, \omega_i)})^{r_i}$ and $z_{i2} = u_i^{r_i}$ and upon receiving pseudonym α_i, node n_i generates $z_{i1} = \alpha_i^{z_i}$ and $z_{i2} = \alpha_i^{t_i}$.

While incorporating the generic Sphinx design, only the computation of the pseudonym α_i changes in the above certificateless key agreement protocol. As above, the pseudonym α_i is generated as $\alpha_i = \alpha_{i-1}^{h_b(\alpha_{i-1}, s_{i-1})}$.

4 Performance Comparison

In this section, we compare the performance of the Sphinx-based circuit constructions of the three protocols with their original constructions.

Message sizes. Message compactness is an important advantage of using Sphinx. The major savings in the length of a circuit construction message comes from reuse of a pseudonym to which blinding is added at each circuit node.

Following the Sphinx notation, p is the size of a public key element in group $\mathbb{G}$ and r is the maximum length of the circuit. We aim at $\kappa = 128$-bit security and use the elliptic curve (ECC) setting with points (compressed form) of size $p = 256$ bits, such as provided by Dan Bernstein's Curve 25519 [2] used by Sphinx. For the finite field setting ($\mathbb{F}$), as higher values amplify our advantage, we consider a DH modulus of size just $p = 2048$ bits to model 128-bit security. To mitigate a recent attack on Tor by Evans, Dingledine and Grothoff [10], the maximum circuit length for recent versions of Tor is set as 8. Therefore, we set $r = 8$ for our Sphinx-based design. However, while comparing, we give an advantage to the other protocols by using Tor's default circuit size $\nu = 3$ for them; using $r = 3$ in our design will only increase our advantage. Additionally, see [17] for a discussion of the effect of Tor's "CREATE_FAST" mechanism.

In the Sphinx-based OR construction, the user sends the tuple $(\alpha_0, \beta_0, \gamma_0)$ to node n_0. The lengths of the elements in this tuple are p, $(2r-1)\kappa$ and κ respectively. The total length, therefore, is equal to $p + 2r\kappa$. In the chosen ECC setting, this is equal to 1280 bits, while for the chosen finite field setting, this is equal to 3072 bits. The message size does not depend upon a specific OR design.

In the original Tor-preDH, PB-OR and CL-OR protocols, this cost is equal to $r(p + 2\kappa)$ as each layer of onion in those constructions requires p bits for a pseudonym, κ bits for identity of the nodes and κ bits for message integrity. With $\kappa = 128$ and $\nu = 3$, this length is equal to 1536 bits in the ECC setting

Table 1. Comparison between lengths (in bits) of various single-pass OR circuit construction messages for 128-bit security ($\kappa = 128$)

Scheme	**Circuit Size**	**UC Security**	**Message Size (bits)**	$\mathbb{F}$ ($p = 2048$)	**ECC** ($p = 256$)
ØS07 [17]	$\nu = 3$	×	$\nu(p + 2\kappa)$	6912	1536
PBOR [15]	$\nu = 3$	×	$\nu(p + 2\kappa)$	–[a]	1536
CL-OR [6]	$\nu = 3$	×	$\nu(p + 2\kappa)$	6912	1536
CL05 [4]	$\nu = 3$	√	$\nu(p + \kappa)$	6528	1920[b]
Sphinx-OR	$r = 8$	√	$\mathbf{p + 2r\kappa}$	**3072**	**1280**

[a] With the necessity of pairings in the PB-OR protocol, we do not consider the finite field setting for it. [b] As we use an Elgamal ciphertext in ECC, $p' = 2p = 512$.

and 6912 bits in the finite field setting. These values are significantly larger than those in our generic format that can make circuits of any length up to 8.

We also consider Camenisch and Lysyanskaya's design in [4] that is secure in the UC model. The message length there is $r(p + \kappa)$. In the ECC computation, we use an Elgamal ciphertext of two $\mathbb{G}$ elements of length $p' = 2p = 512$ instead of p. For $\nu = 3$, the message sizes are 1920 bits and 6528 bits respectively. Therefore, our Sphinx-based design achieves the same security guarantees with much smaller messages. Table 1 provides a succinct representation of the above discussion. Note that as Tor generates a circuit in a telescoping form, we do not compare it with the single-pass protocols.

Computational cost. Compact messages and security in the UC model do not come without some additional computational cost. However, importantly, there is no addition to the computations done by users (possibly hundreds of thousands of them), while the increase is easily manageable for OR nodes. Each node in a circuit has to perform an additional exponentiation in $\mathbb{G}$ as it prepares the pseudonym for the next node. However, one exponentiation in $\mathbb{G}$ costs around 1 ms on a desktop machine. This does not affect the overall circuit construction cost in practice, which is in seconds due to the network latency.

Comparison between the three OR constructions. In our full version [14], we also compare the Tor-preDH, PB-OR and CL-OR protocols with Tor as well as with each other in terms of their computational and infrastructural costs. We observed that in multi-pass constructions, Tor-preDH is the most efficient. However, in the absence of a clearly optimal scheme, the choice among the single-pass circuit constructions has to be made based on the size of a prospective anonymity network and availability of a PKG infrastructure. For smaller networks, Tor-preDH and CL-OR are better suited than PB-OR. However, the choice between those two is tricky. In Tor-preDH, the directory server and users have to verify OR nodes' public key certificates once per forward secrecy phase. In CL-OR, for every circuit construction of length ν a user has to perform ν additional exponentiations and every circuit node has to perform one additional exponentiation. For large anonymity networks, we find PB-OR to be more usable. The public-key downloads saved there are more than compensate for the infrastructure cost incurred by a (distributed) PKG. Further, using the CLC setting, it

may be possible to avoid the public-key scalability and key escrow issues at the same time and it is an interesting future work to design such a scheme.

Acknowledgements. We thank D. Fiore for providing the camera-ready version of his certificateless onion routing paper [6] with D. Catalano and R. Gennaro. We also thank R. Dingledine, G. Zaverucha, and the anonymous reviewers for providing valuable feedback. This work is supported by NSERC, MITACS, and a David R. Cheriton Graduate Scholarship.

References

1. S. S. Al-Riyami and K. G. Paterson. Certificateless Public Key Cryptography. In *Advances in Cryptology—ASIACRYPT'03*, pages 452–473, 2003.
2. D. J. Bernstein. Curve25519: New Diffie-Hellman Speed Records. In *Public Key Cryptography (PKC'06)*, pages 207–228, 2006.
3. D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. In *Advances in Cryptology—CRYPTO'01*, pages 213–229, 2001.
4. J. Camenisch and A. Lysyanskaya. A Formal Treatment of Onion Routing. In *Advances in Cryptology—CRYPTO'05*, pages 169–187, 2005.
5. R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *FOCS'01*, pages 136–145, 2001.
6. D. Catalano, D Fiore, and R. Gennaro. Certificateless Onion Routing. In *CCS'09*, pages 151–160, 2009.
7. W. Dai. PipeNet 1.1. `http://www.weidai.com/pipenet.txt`, 1998. Accessed Nov. 2009.
8. G. Danezis and I. Goldberg. Sphinx: A Compact and Provably Secure Mix Format. In *IEEE Symposium on Security and Privacy*, pages 269–282, 2009.
9. R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *13th USENIX Security Symposium*, pages 303–320, 2004.
10. N. S. Evans, R. Dingledine, and C. Grothoff. A Practical Congestion Attack on Tor Using Long Paths. In *18th USENIX Security Symposium*, pages 33–50, 2009.
11. M. J. Freedman and R. Morris. Tarzan: A Peer-to-Peer Anonymizing Network Layer. In *CCS'02*, pages 193–206. ACM, 2002.
12. S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.
13. D. M. Goldschlag, M. Reed, and P. Syverson. Hiding Routing Information. In *Information Hiding: First International Workshop*, pages 137–150, 1996.
14. A. Kate and I. Goldberg. Using Sphinx to Improve Onion Routing Circuit Construction. Technical Report CACR 2009-33, 2009. Available at `http://www.cacr.math.uwaterloo.ca/techreports/2009/cacr2009-33.pdf`.
15. A. Kate, G. M. Zaverucha, and I. Goldberg. Pairing-Based Onion Routing. In *PETS'07*, pages 95–112, 2007.
16. A. Menezes, P. Van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1st edition, 1997.
17. L. Øverlier and P. Syverson. Improving Efficiency and Simplicity of Tor Circuit Establishment and Hidden Services. In *PETS'07*, pages 134–152, 2007.
18. M. Reed, P. Syverson, and D. Goldschlag. Anonymous Connections and Onion Routing. *IEEE J-SAC*, 16(4):482–494, 1998.
19. R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems Based on Pairing. In *Symposium on Cryptography and Information Security (SCIS'00)*, Japan, 2000.
20. The Tor Project. . `https://www.torproject.org/`, 2003. Accessed Nov. 2009.