Fully Non-interactive Onion Routing with Forward-Secrecy

Dario Catalano[1], Mario Di Raimondo[1], Dario Fiore[2,⋆], Rosario Gennaro[3], and Orazio Puglisi[1]

[1] Dipartimento di Matematica ed Informatica – Università di Catania, Italy
{catalano,diraimondo}@dmi.unict.it, puglisi.o@gmail.com
[2] École Normale Supérieure, CNRS - INRIA, Paris, France
dario.fiore@ens.fr
[3] IBM T.J. Watson Research Center. Hawthorne, New York 10532.
rosario@us.ibm.com

Abstract. In this paper we put forward a new onion routing protocol which achieves forward secrecy in a *fully non-interactive* fashion, without requiring any communication from the router and/or the users and the service provider to update time-related keys. We compare this to TOR which requires $O(n^2)$ rounds of interaction to establish a circuit of size n. In terms of the computational effort required to the parties, our protocol is comparable to TOR, but the network latency associated with TOR's high round complexity ends up dominating the running time. Compared to other recently proposed alternative to TOR (such as the PB-OR and CL-OR protocols) our scheme still has the advantage of being non-interactive (both PB-OR and CL-OR require some interaction to update time-sensitive information), and achieves similar computational performances. We performed extensive implementation and simulation tests that confirm our theoretical analysis. Additionally, while comparing our scheme to PB-OR, we discovered a flaw in the security of that scheme which we repair in this paper.

Our solution is based on the application of forward-secure encryption. We design a forward-secure encryption scheme (of independent interest) to be used as the main encryption scheme in an onion routing protocol.

1 Introduction

As we move to use network communication in more and more aspects of everyday's life, it has become apparent that our privacy is at stake. The ability to monitor electronic communication, to store large amount of data, and to run sophisticated analytics on it, allows a sufficiently motivated party to "connect the dots" between various on-line activities of a specific user and get a pretty accurate picture of his/her private life.

These privacy concerns were recognized since the beginning of the Internet age, and *anonymous communication* was conceived as a possible approach to

⋆ Work partially done while student at University of Catania.

J. Lopez and G. Tsudik (Eds.): ACNS 2011, LNCS 6715, pp. 255–273, 2011.

their solutions. Anonymity is the user's ability to hide not only her identity but also her network information (e.g. her network address). This is of utter importance in many real life applications, where a user's identity should be decoupled from her network activities (e.g. voting, e-cash, anonymous credentials, etc.).

Chaum in 1981 [8], proposed the notion of a *anonymous channel*, realized through a *mix-net*: very informally, his idea was to route messages through a series of nodes (the mix-net). The messages are "wrapped" in several layers of encryptions and sent to the first node in the mix-net. Each node, batches a number of received ciphertexts, peels off a layer of encryption from each of them, and sends the resulting values in permuted order to the next node. The last node in the mix-net delivers the messages to the recipients. Anonymity derives from the fact that since each node permutes the messages in a randomized order before forwarding, no traffic analysis can actually link the sender to the receiver.

Goldschlag *et al.* introduced in [17] the so-called *Onion Routing* approach which is based on Chaum's idea as follows. Consider a setting defined by: a service provider, a set of users and a set of nodes (called *onion routers*). The user's goal is to establish an anonymous channel that allows him to send messages over the network without being identified. In order to do so, he selects a random set of nodes (called a *circuit*), wraps the message with several layers of encryption, one for each node, and sends it through these intermediate nodes. Because of their layered composition such wrapped messages are called *onions*. Whenever a node receives a message, it decrypts it and immediately sends the resulting value to the next node. Note that differently from Chaum's mixers, an onion router does not collect and permute a batch of messages before forwarding, but it immediately forwards what it receives. Roughly speaking, since the user chooses a random subset of the routers to forward his messages, anonymity is guaranteed by the assumption that the adversary cannot monitor the entire set of onion routers, but has only a *local* view of the network communication. This very simple and elegant idea has proven itself very popular over the Internet. Besides leading to several other constructions and implementations (e.g. [17,9,25,18,15,26,14]), it gave birth to the Onion Routing Project, later replaced by Tor [14] (the second generation onion router) which provides privacy and anonymity to a large number of users over the Internet. At the moment it counts, roughly, 1000 onion routers and hundreds of thousands of users over the world.

An important aspect of onion routing protocols is how messages travel securely through each onion router. The idea given in [17] proposes that the user encrypts a random symmetric key with the public key of each onion router: the symmetric key is used to encrypt the corresponding onion layer and the name of the next node in the circuit. This approach, unfortunately, is not robust in the face of possible server corruptions. Indeed if an adversary obtains the long-term secret key of a router O, it can then decrypt *all* the ciphertexts received by O: particularly troubling is that the adversary can decrypt communication that happened *before* the leakage of the secret key. Resistance to such attacks has been already recognized as an important security property (that is called *forward secrecy*) in other cryptographic contexts. So, even for onion routing, ideally we would like to have a protocol that is *forward-secure*, namely such that a router's corruption does not reveal anything about communication prior to the corruption.

In the context of Tor, Dingledine *et al.* [14] proposed a solution which relies on using the routers' public keys only to establish a temporary session key via an (interactive) Diffie-Hellman [12] key agreement. In order to get anonymity such interaction is made part of a specific protocol called *Tor Authentication Protocol* (TAP) which was proven secure by Goldberg [16]. The main idea of TAP is the *telescoping* technique that allows to construct the circuit and to exchange the temporary keys anonymously. However, this technique has a major drawback: its bandwidth and round complexity. In fact, in order to build a circuit of length n it is required to exchange $O(n^2)$ (symmetrically encrypted) messages. Øverlier and Syverson [24] later improved the efficiency of TAP by proposing the use of only half-certified Diffie-Hellman key-agreement, but the round complexity of telescoping is still quadratic.

A related notion of forward secrecy (sometimes called *eventual forward secrecy* in the literature) can be realized by frequently changing the long-term server keys, in order to minimize the security impact of key leakage. If the adversary learns the secret key of a server O, it may only learn the communication related to the validity period of that key. The trivial implementation of this idea (changing the servers public keys) would be very complicated in practice as it forces the routers to generate new keys with corresponding valid certificates, and the users to repeatedly obtain such certified keys.

Recently, two approaches were proposed to achieve eventual forward secrecy in a more efficient and simple way. In 2007, Kate *et al.* [20,21] proposed using identity-based encryption schemes such as [4,29] to construct an onion routing protocol called PB-OR (for pairing-based onion routing). In identity-based cryptography (introduced by Adi Shamir in [27]) the parties' public keys are their identities, and the secret keys are provided to them by a trusted Key Generation Center (KGC). PB-OR uses the original onion-routing idea to encrypt messages using the public key of the routers, except that in this case the routers' public keys are their identities together with the validity period. Therefore a router's corruption reveals only the messages encrypted during the particular period of the corruption. PB-OR has two major drawbacks: (i) the existence of a trusted KGC who can decrypt any message in the network; (ii) it requires the routers to interact with the KGC at each validity period to obtain new secret keys. While the former can be solved by using known techniques (e.g. a distributed KGC), the latter is more annoying and seems to be inherent in that construction.

These two drawbacks were addressed in a subsequent paper by Catalano *et al.* [7] who suggested using *certificateless* encryption (rather than identity-based) to construct the onions. Certificateless encryption [1] is an hybrid setting that lies between public key and identity-based cryptography: each user has an identity string ID with a matching secret key produced by a KGC and also a public/secret key pair, as in the traditional public key model but with the advantage that such key needs *not* be certified. Certificateless encryption does not suffer the problem of key escrow as the KGC cannot decrypt the messages sent to a user. The CL-OR protocol in [7] achieves eventual forward secrecy by having the routers periodically change their public keys: compared to PB-OR, CL-OR requires the users to interact with the service provider at each time period to obtain the routers' new public keys (but with the advantage of not having to manage and verify certificates).

Our Contribution. This paper presents a new onion routing protocol which outperforms TOR (and the other proposals such as PB-OR and CL-OR as well). The main improvement of our proposal is that it is *fully non-interactive.* Our main idea is to achieve eventual forward secrecy by using *forward-secure identity-based encryption (fs-IBE)* for the public keys of the routers.

Forward-secure public-key encryption (fs-PKE) was originally proposed by Anderson in [2] exactly as a way to achieve eventual forward secrecy for public-key encryption, without requiring users to continuously change their public keys. In Anderson's idea, a user U of a fs-PKE scheme publishes a static public key pk_U and senders encrypt messages using this public key and a time value t. To decrypt such ciphertexts, U uses a secret key $sk_{U,t}$. At the beginning, U holds the secret key $sk_{U,0}$, and at each time period U *updates* its secret key from $sk_{U,t}$ to $sk_{U,t+1}$ and erases $sk_{U,t}$. This process must be one-way – while it is easy to compute $sk_{U,t+1}$ from $sk_{U,t}$ the reverse must be hard – in order to achieve eventual forward secrecy: if $sk_{U,t}$ is compromised past communication remains secret. Our contribution can be described as follows:

- First, we propose a new onion routing protocol *fs-ID-OR*, which uses the "classical" onion-routing approach to construct onions by using the static public keys of the routers, except that we use an identity-based forward secure encryption (*fs-IBE*) scheme for the routers' public-keys.
- Next, we build an fs-IBE scheme by carefully applying a generic paradigm by Canetti *et al.* [6] to the Hierarchical Identity-Based Encryption of Boneh *et al.* [3]. This scheme is tailored to the onion-routing application and has other properties (discussed below) which can make it of independent interest.

The advantages of fs-ID-OR compared to PB-OR and CL-OR are substantial. Compared to PB-OR, our new scheme does not require the KGC to be involved in the generation of new secret keys for the routers: indeed in fs-ID-OR the update of the secret key at each time period is a local, non-interactive procedure performed by the router. Compared to CL-OR the public keys of the routers are fixed throughout all time periods (only the secret keys change) so the users do not need to obtain new public keys for the routers after each time period. Compared to Tor, fs-ID-OR has a completely non-interactive circuit-building protocol with linear round complexity. This makes fs-ID-OR a *truly non-interactive solution* as it requires no interaction between the routers, the KGC or the users to update time information after each time change.

It is fair to notice that in practice clients have to receive up-to-date informations about the state of the network to ensure that they create correct circuits (e.g. restrictions on the paths, status of online nodes, delays, etc.) since these informations are all security sensitive items. Therefore a truly secure solution seems to be interactive anyway, and the advantage of our proposal limited. However, we argue that the cost of exchanging and processing cryptographic information related to the protocol is likely to be orders of magnitude larger than the cost of receiving network status updates, and therefore removing interaction from the cryptographic part of the protocol is not just a theoretical exercise, but a real practical advantage.

Note that the level of protection afforded by eventual forward-security is related to the frequency of updates of the long-term keys (more frequent updates imply less past information being leaked in case of a key compromise). Because our solution is non-interactive, we have removed the major cost of key updates, thus making very frequent updates possible. Remarkably, our non-interactive solution does not come with high efficiency cost. In terms of computational load, our protocol is comparable with PB-OR, and definitely better than Tor (which is saddled by the cost of telescoping). The performance details of our protocol are discussed in Section 6 where we report an extensive implementation and simulation tests. The basic version of our protocol works in the identity-based setting: therefore we must assume a trusted KGC who has the ability to decrypt all communications, as in PB-OR. However, we stress on that it is possible to modify our protocol to work in both the classical PKI setting and in the certificateless setting so to avoid the key-escrow problem. We discuss these variations in Section 4.

Note. While proving the security of our scheme, we noticed a small flaw in one of the security arguments in [21]. They claim that the anonymity property can be achieved by assuming that the encryption schemes used in the Onion Routing protocol are simply semantically secure. But our proof of security shows that anonymity relies in a crucial way on assuming that the encryption schemes are secure against chosen ciphertext attack.

Other Related work. We refer the reader to the work in [23,5] for formal security definitions for the problem of onion routing. We discuss the relationship of our work with respect to the Camenisch and Lysyanskaya formal model [5] in Section 2. A forward-secure (hierarchical) identity-based public key encryption is presented by Yao *et al.* in [30]. Our new scheme is somewhat uncomparable to theirs: our scheme was designed to satisfy only the minimal security properties needed for the onion-routing application, while the scheme in [30] proposes additional security properties which might be useful in other contexts. As a result our scheme is simpler and more efficient (in particular it allows for constant size ciphertexts), but does not satisfy all the security properties proposed in [30].

2 Forward-Secure Identity-Based Onion Routing

In this section we introduce the notion of *Forward-Secure Identity-Based Onion Routing* (*fs-ID-OR*). As usual, an onion routing protocol is characterized by a service provider, a set of "onion routers" and users. The goal of the protocol is to provide anonymity over a network to users and the basic idea is that users route their traffic throughout an encrypted circuit of randomly chosen onion routers.

In addition, our solution considers *forward-secrecy*, a property which is in general quite important for cryptographic protocols. Informally, it guarantees that the security properties still hold even if the adversary can corrupt *all* the parties and learn their secret keys *after* a protocol session is expired. If one focuses only on adversaries that can corrupt parties *after* a specific time period τ, then we call this property *eventual* forward-secrecy. Otherwise it is called *immediate* forward-secrecy. In our work we will focus on eventual forward-secrecy since it is the strongest notion that is achievable in a non-interactive way. Our definition of fs-ID-OR follows the traditional notion of onion routing but focuses on the identity-based setting where each onion router is represented by a unique identity string OR_i (e.g. its name, address, etc.) and receives a secret key related to such string by a trusted entity, called the *Key Generation Center* (KGC).

In order to consider forward-secrecy, we assume that the time is split into time periods of the same length. A fs-ID-OR protocol consists of the following phases:

Setup and Key Generation. The service provider generates the global parameters of the system and makes them available to all users. When an onion router with identity OR_i joins the system at time t, the service provider acts as a KGC and uses its master secret to generate a private key $\mathsf{sk}_{OR_i,t}$ for OR_i. When a time period t expires, each onion router is required to update its secret key, that means that it runs a specific algorithm that on input $\mathsf{sk}_{OR_i,t}$ outputs $\mathsf{sk}_{OR_i,t+1}$ while the old key is erased.

Circuit construction. In this phase the user firstly has to obtain a list L containing the identities of all the available onion routers. Such list is maintained by the KGC and is updated at a specific interval t'. We notice that t' might also be different from the τ used for updating the keys. Next, the user chooses an ordered set of n onion routers $OR_1, \ldots, OR_n$ at random among those in L. This ordered set is called *circuit* and the number n is typically fixed and specified in the protocol parameters. In order to send messages through the circuit at time t, the user builds a special ciphertext O_1, called "onion ciphertext" such that, for all $i = 1$ to n onion router OR_i is able to partially decrypt the onion O_i (using its secret key of time t). From such partial decryption it obtains: (i) the address of the next router OR_{i+1} in the circuit (ii) and another onion ciphertext O_{i+1}. The user sends O_1 to OR_1 and whenever router OR_i receives O_i, it decrypts it and forwards O_{i+1} to OR_{i+1}. Finally, the last router of the circuit gets the message m and the address P of the actual recipient, and forwards m to P.

2.1 Security of Forward-Secure Identity-Based Onion Routing

Now we describe the security properties that a fs-ID-OR protocol should satisfy.

Integrity and Correctness. *Correctness* states that when parties follow the protocol, then the recipient should get the message that was originally sent and encrypted by the sender. Let n be a pre-specified upper bound for the number of routers in a circuit. Then we say that an onion routing protocol satisfies *integrity* if it is possible to recognize an onion ciphertext which is intended for more than n routers.

Cryptographic Unlinkability. Cryptographic unlinkability formalizes in a cryptographic way the fact that a fs-ID-OR protocol provides anonymity. Informally, this property says that an attacker should not be able to find a link between the sender and the receiver of a given communication. We point out, as explained in [21], that network-level attacks are not considered at this stage.

Consider the following game between an adversary $\mathcal{A}$ and a Challenger:

Setup. The Challenger generates the public parameters and gives them to $\mathcal{A}$.

Phase 1. In this phase the adversary is allowed to:
- corrupt onion routers and learn their secret keys (at specific time t);
- submit a tuple (OR, t, O) to get the decryption of O under OR's secret key at time t.

Challenge. At some point the adversary is allowed to choose a message m, a time period t^* and routers OR_1, OR'_1, OR_2, OR'_2, OR_H such that OR_H is honest (i.e. OR_H has not been corrupted in the previous phase, or it has been corrupted at time $t > t^*$). The Challenger flips a binary coin $b \xleftarrow{\$} \{0, 1\}$ and proceeds as follows. If $b = 0$ it creates:
- O_1 as the onion for the circuit (OR_1, OR_H, OR_2)
- and O'_1 as the onion for the circuit (OR'_1, OR_H, OR'_2).

Otherwise, if $b = 1$ it creates
- O_1 as the onion for the circuit (OR_1, OR_H, OR'_2)
- and O'_1 as the onion for the circuit (OR'_1, OR_H, OR_2).

Let O_H, O_2 and O'_H, O'_2 be the onion ciphertexts contained into O_1 and O'_1 respectively. Finally (O_1, O'_1) is given to the adversary.

Phase 2. $\mathcal{A}$ can proceed as in Phase 1 except that:
- OR_H cannot be corrupted at time $t \leq t^*$;
- $\mathcal{A}$ cannot submit (OR_H, t^*, O_H) and (OR_H, t^*, O'_H) to the decryption oracle (otherwise the adversary would trivially win);
- $\mathcal{A}$ can ask to the Challenger the decryption of a pair (O, O') under OR_H's secret key at time t^*. However in this case the Challenger does the following. It first decrypts O and O' and gets $(\overline{OR}, \overline{O})$ and $(\overline{OR'}, \overline{O'})$ respectively. If $\overline{OR} = OR_2$ and $\overline{OR'} = OR'_2$ then the Challenger outputs $((\overline{OR}, \overline{O}), (\overline{OR'}, \overline{O'}))$. Otherwise if $\overline{OR} = OR'_2$ or $\overline{OR'} = OR_2$ then $\mathcal{A}$ is given $((\overline{OR'}, \overline{O'}), (\overline{OR}, \overline{O}))$ (i.e. the tuple corresponding to OR_2 is always given first). We notice that such a requirement is essential, otherwise the adversary might trivially win the game.

Guess. At the end, the adversary outputs a guess b' for b and it wins if $b' = b$.

We define the advantage of an adversary $\mathcal{A}$ in this game as $\mathbf{Adv}^{anon}_{ID-OR}(\mathcal{A}) = 2\Pr[b' = b] - 1$ and we say that an onion routing protocol has cryptographic unlinkability if for any PPT adversary $\mathcal{A}$, $\mathcal{A}$'s advantage is at most negligible.

Remark. A more general definition would consider an adversary $\mathcal{A}$ that in the challenge phase can choose circuits of length n instead of length 3. However, since we assume that all but one (i.e. OR_H) routers can be corrupted, one may think to OR_1 (resp. OR_2) as the collapsed set of adversarially controlled routers before (resp. after) the single honest one (i.e. OR_H), and the same can be done for OR'_1, OR_2 and OR'_2.

Circuit Position Secrecy. This property says that it should not be possible to learn a router's position in the circuit by looking at the ciphertext it is receiving. In those constructions where the onion ciphertexts are built as re-encryptions with several keys (e.g. $\Gamma = E_{K_1}(E_{K_2}(\cdots E_{K_n}(m)\cdots)))$ this property cannot hold. Camenisch and Lysyanskaya [5] showed that it is in fact sufficient to look at the ciphertext's size to derive such information. Indeed, in every randomized encryption scheme the ciphertext space is bigger than the plaintext one (typically by a constant). However solutions to this problem have been proposed [5,21,11,19].

3 A Generic Construction of FS-ID Onion Routing

In this section we show how to construct a fs-ID-OR protocol in a black-box way from any forward-secure identity-based key encapsulation mechanism (fs-IB-KEM) and a symmetric encryption scheme.

Our construction generalizes the idea of Kate *et al.* [21] whose scheme can be seen as an instantiation of our generic construction when using the IB-KEM of Boneh and Franklin [4] and considering an interactive protocol for updating routers' keys (i.e. the KGC generates new keys every time period).

In what follows we define the primitives that are relevant for our construction.

Forward-Secure Identity-Based Key Encapsulation. A Forward-Secure Identity-Based Key Encapsulation Mechanism (fs-IB-KEM) is defined by the following algorithms:

$\mathsf{Setup}(1^k, T)$**.** It takes as input the security parameter k and the total number of time periods T and outputs a public key MPK and a master secret key MSK.

$\mathsf{KeyGen}(MSK, \mathsf{ID}, t)$**.** The key generation algorithm uses the master secret key to produce a private key $\mathsf{sk}_{\mathsf{ID},t}$ that is related to the identity ID and the time period t.

$\mathsf{KeyUpdate}(\mathsf{sk}_{\mathsf{ID},t})$**.** Given in input $\mathsf{sk}_{\mathsf{ID},t}$ (the secret key of identity ID for time period t) the key update algorithm outputs a new key for time period $t+1$.

$\mathsf{Encap}(MPK, \mathsf{ID}, t)$**.** Given the master public key, an identity ID and a time period t, the encapsulation algorithm outputs a ciphertext C and a session key K.

$\mathsf{Decap}(\mathsf{sk}_{\mathsf{ID},t}, C)$**.** The decapsulation algorithm uses the secret key of identity ID and time period t to recover the session key K from a ciphertext C.

Correctness requires that for all identities $\mathsf{ID} \in \{0,1\}^*$ and time periods $0 \leq t < T$:

$$\Pr\left[\begin{array}{l}(MPK, MSK) \xleftarrow{\$} \mathsf{Setup}(1^k, T); \mathsf{sk}_{\mathsf{ID},t} \xleftarrow{\$} \mathsf{KeyGen}(MSK, \mathsf{ID}, t);\\ (C, K) \xleftarrow{\$} \mathsf{Encap}(MPK, \mathsf{ID}, t); K' \leftarrow \mathsf{Decap}(\mathsf{sk}_{\mathsf{ID},t}, C) : K' = K\end{array}\right] = 1$$

We notice that the same holds when the secret key $\mathsf{sk}_{\mathsf{ID},t}$ is obtained via the key update algorithm. Below we define the notion of forward-security against chosen-ciphertext attacks (fs-ID-IND-CCA) for fs-IB-KEM schemes. Consider the following game between an adversary $\mathcal{A}$ and a Challenger:

Setup. A pair of master keys $(MPK, MSK) \xleftarrow{\$} \mathsf{Setup}(1^k, T)$ is generated and the adversary is given MPK.

Phase 1. In this phase the adversary is given access to oracles $\mathsf{breakin}(\cdot,\cdot)$ and $\mathsf{Decap}(\cdot,\cdot,\cdot)$ as follows:
- On input (ID, i) the $\mathsf{breakin}$ oracle computes the secret key $\mathsf{sk}_{\mathsf{ID},i}$ and gives it to the adversary.
- On query (ID, i, C) to the decapsulation oracle, the Challenger computes the key $\mathsf{sk}_{\mathsf{ID},i}$ and gives $K \leftarrow \mathsf{Decap}(\mathsf{sk}_{\mathsf{ID},i}, C)$ to the adversary.

Challenge. At some point the adversary is allowed to output a pair (ID^*, t^*) such that either ID^* is different from all the identities queried to $\mathsf{breakin}$ in the previous phase or $\mathsf{ID}^* = \mathsf{ID}_j$ and $t^* < t_j$ (where (ID_j, t_j) was the j-th query to $\mathsf{breakin}$). The Challenger computes $(C^*, K_0) \xleftarrow{\$} \mathsf{Encap}(MPK, \mathsf{ID}^*, t^*)$ and picks a random session key $K_1 \xleftarrow{\$} \mathcal{K}$. Then it flips a random bit $b \xleftarrow{\$} \{0,1\}$ and gives (C^*, K_b) to the adversary.

Phase 2. As Phase 1 except that the adversary is not allowed to query the decapsulation oracle on $(\mathsf{ID}^*, t^*, C^*)$ and the $\mathsf{breakin}$ oracle on (ID^*, j) with $j \leq t^*$.

Guess. At the end of the game the adversary outputs a bit b' as its guess for b.

We define the advantage of $\mathcal{A}$ in this game as

$$\mathbf{Adv}_{\mathcal{IB}}^{fs-IND-ID-CCA}(\mathcal{A}) = |2\Pr[b = b'] - 1|.$$

Definition 1 (fs-IND-CCA security). *A fs-IB-KEM is* forward-secure against chosen-ciphertext attacks *if for any PPT adversary $\mathcal{A}$ we have: $\mathbf{Adv}_{\mathcal{IB}}^{fs-IND-ID-CCA}(\mathcal{A}) \leq \epsilon$, where ϵ is negligible in the security parameter.*

3.1 Our Generic Construction

Let $\mathcal{IB} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{KeyUpdate}, \mathsf{Encap}, \mathsf{Decap})$ be a fs-IB-KEM and $\mathcal{E} = (KG, E, D)$ be a symmetric encryption scheme (whose notion is quite standard, and is omitted for lack of space). We construct the following protocol:

Setup. In this phase the KGC runs the setup algorithm of $\mathcal{IB}$ to obtain a master public key MPK and a master secret key MSK. The master public key is made available to all users together with informations about the time. We assume the time to be split into time periods of length τ (e.g. τ = one hour) such that when such a period expires, each onion router updates his secret key as explained in the next phase. Moreover, the KGC maintains a list L containing the identities of all the onion routers available at a specific time. Such list is updated at a specific interval τ', which does not have to be necessarily equal to τ.

Key Generation. Whenever an onion router OR_i joins the system, at time t, the KGC generates a secret key $\mathsf{sk}_{OR_i,t} \xleftarrow{\$} \mathsf{KeyGen}(MSK, OR_i, t)$ for it. To achieve forward secrecy, when a time period t expires each onion router OR_i is required to update his secret key by running $\mathsf{sk}_{OR_i,t+1} \leftarrow \mathsf{KeyUpdate}(\mathsf{sk}_{OR_i,t})$ and erasing $\mathsf{sk}_{OR_i,t}$ from its memory.

Circuit construction. Assume that a user wants to build a circuit at time t. First he obtains the updated list L from the KGC and then he chooses an ordered sequence of n onion routers $OR_1, \ldots, OR_n$ at random among those in L. Next, for all $i = n$ to 1 it proceeds as follows. It runs $(C_i, K_i) \xleftarrow{\$} \mathsf{Encap}(MPK, OR_i, t)$ and creates "onion ciphertext" $O_i = (C_i, \Gamma_i)$ where $\Gamma_i = E_{K_i}(OR_{i+1}, O_{i+1})$. The user sends O_1 to the first onion router in the circuit. Whenever onion router OR_i gets a pair $O_i = (C_i, \Gamma_i)$ it recovers $K_i \leftarrow \mathsf{Decap}(\mathsf{sk}_{OR_i,t}, C_i)$ and then runs $(OR_{i+1}, O_{i+1}) \leftarrow D_{K_i}(\Gamma_i)$. Finally it sends O_{i+1} to OR_{i+1} (which is the next router of the circuit).

The first time a user is using a circuit, he wants to be aware that all the chosen routers are available. Therefore it sends a special message $\perp$ through the circuit (i.e. $\Gamma_n = E_{K_n}(\perp)$). When an onion router decrypts and obtains $\perp$ it learns that it is the last router of the circuit and sends back a confirmation message $E_{K_n}(Ack)$ to the previous router. Upon the receipt of a confirmation message an onion router OR_i encrypts it using K_i and sends it to the previous router. For this reason, we assume that each router keeps in memory a session state containing the two adjacent nodes and the session keys. We notice that this is also useful to prevent replay attacks. Finally, upon the receipt of a confirmation message, the user verifies its validity by decrypting it using the session keys $K_1, \ldots, K_n$.

Once the circuit has been successfully established, the user will use it to send messages over the network. In particular, he will re-use the same session keys $K_1, \ldots, K_n$ to form the onions. This allows to avoid expensive asymmetric encryption (and decryption) operations.

3.2 Security

Integrity and Correctness. Let n be the fixed upper bound for the number of routers in the circuit. We notice that an onion ciphertext containing more than n layers of encryption can be easily recognized by looking at its length. Therefore our protocol has integrity. On the other hand correctness easily follows from the construction and the correctness of the two employed encryption schemes.

Cryptographic Unlinkability. The property is proven by the following theorem whose proof is omitted for lack of space.

Theorem 1. *If $\mathcal{IB}$ is fs-ID-IND-CCA secure and $\mathcal{E}$ is IND-CCA secure, then the protocol given in Section 3 satisfies cryptographic unlinkability.*

A remark on cryptographic unlinkability. The previous theorem proves the cryptographic unlinkability of our generic construction by assuming that both the fs-IB-KEM and the symmetric encryption schemes are secure in an IND-CCA sense. We need this property because of the adversary's (realistic) ability to ask decryption of onions (e.g. he may simply send an onion to a router and look for its outgoing packets).

Cryptographic unlinkability was first defined in [21] (though in a slightly less formal way). There the authors claimed that for their construction this property is implied by the IND-CPA security of the symmetric encryption scheme. The proof of this claim is not formal and it is unclear how the proof can manage the adversary's decryption queries in the "C processing" phase.

Moreover, we notice that if only IND-CPA security is considered, then the adversary might modify only one of the two challenge ciphertexts O_H, O'_H in such a way that, after seeing their decryptions, it can recognize which of the two onions they come from. More precisely, $\mathcal{A}$ (who owns all secret keys but OR_H's) may keep O'_H the same and modify O_H such that the encrypted onion will decrypt to a random message[1]. When $\mathcal{A}$ later receives the two decrypted onions, it will be able to recognize what was the path chosen by the challenger. On the other hand, in our case assuming IND-CCA security allows to obtain a correct and formal proof of cryptographic unlinkability.

Circuit Position Secrecy. Unfortunately, our protocol does not satisfy this property as it is vulnerable to the attack showed by Camenisch and Lysyanskaya in [5] that allows to learn the circuit's position of a ciphertext's recipient. Precisely, this can be done by looking at the length of a ciphertext. However, if one is interested into this property, then it is possible to slightly modify our protocol using the technique proposed by Kate *et al.* in [21]. Its application to our protocol is straightforward and thus we can obtain a protocol with circuit position secrecy, even if this comes at the cost of having longer ciphertexts.

4 Certificateless and PKI Variants

The onion routing scheme we presented in Section 3, uses an *identity-based* forward-secure encryption for the routers. This means that the routers' identities serve as their public keys and the secret keys are provided to them by a trusted KGC. We chose this approach to minimize the size of the public information required to run the system: public keys and certificates (users need to know only the KGC's). The obvious drawback of this approach is *key escrow:* the KGC has the ability to decrypt any message. In this section we describe two simple variations to eliminate the key escrow problem from our scheme. One will yield a scheme in the classical PKI setting: each router has his own public key and certificate. The second variation will be a certificateless (CL) scheme: in this case together with the KGC's keys, routers will hold public keys which however need not be certified. Both variations pay some price compared to the identity-based scheme presented earlier: when instantiated with our scheme of Section 5,

[1] The definition of IND-CPA security does not rule out that this is possible, and indeed it can be done in many IND-CPA secure schemes.

the PKI version has to face long public keys (but no increase in computation); the CL one requires a few extra exponentiations to the user. Details follow.

A PKI Variation. To obtain eventual forward secrecy for an onion-routing protocol, it is sufficient to use any forward-secure encryption scheme, not necessarily an *identity-based* one. In particular, one could use our scheme where each router acts as his own KGC, and give himself different keys for each time period. If we were to follow this approach, there would be no centralized KGC and no key escrow problem. To create an onion, a user would have to do the same amount of work as in the identity-based scheme above. The only problem is that the concrete scheme we propose in Section 5 has longer public keys. Thus the amount of data to be stored at each user would be large.

A Certificateless Variation. There is a generic way to transform any ID-based encryption into a CL one [1]. The receiver R, who already holds a secret key sk_R related to his identity and provided to him by the KGC, also publishes an independent public key PK and keeps the secret key SK. To encrypt a message m for R, a sender splits $m = m_1 \oplus m_2$, with m_1 random, and sends m_1 encrypted with the ID-based scheme, and m_2 encrypted under PK. As pointed out in [1] the public key PK needs not be certified to belong to R (intuitively this is because only R can decrypt m_1). The advantage is that now the KGC cannot decrypt the message m. This generic paradigm can be efficiently implemented in our case. In our protocol the user establishes a shared symmetric key k_i with the i^{th} router in the circuit using the id-based KEM described in the previous section. The key k_i is used to encrypt the i^{th} layer of the onion. To transform this scheme into a CL one, each router can publish a public key and the user runs another KEM to establish another key k_i' with it, and the i^{th} layer of the onion is encrypted with $k_i \oplus k_i'$. An efficient instantiation of this KEM could be any KEM that works over the same cyclic group used for the ID-based scheme (so that no new public information must be generated), e.g. establishing a random key using ElGamal. This approach requires the user to compute n extra exponentiations to create an onion (n is again the length of the circuit).

5 The Proposed Construction

In this section we present a concrete scheme that realizes a fs-IB-KEM with $T = 2^{\ell+1} - 1$ time periods. Our solution is presented in two steps. First, we give a forward secure identity based encryption scheme (fs-IBE) that is probably secure only in an IND-CPA sense. Next, we apply a simple variant of Dent's transformation [10] in order to convert our basic fs-IBE into an IND-CCA secure fs-IB-KEM.

As for the first construction, we construct our fs-IBE as follows. Following the idea of Canetti-Halevi-Katz [6], we use a binary tree of height ℓ where each time period is associated with a node of the tree according to a pre-order traversal. If w^t is the node of the tree associated with time period t we have:

– $w^0 = \epsilon$, i.e. the root of the tree;
– if w^i is an internal node, then $w^{i+1} = w^i||0$ (where $||$ is the concatenation operator);
– if w^i is a leaf node (and $i < T - 1$) then $w^{i+1} = w'1$ where w' is the longest string such that $w'0$ is a prefix of w^i.

The proposed scheme builds upon the HIBE of Boneh, Boyen and Goh [3] and the generic construction of Canetti-Halevi-Katz [6] as follows. The first level of the hierarchy contains the identities and then each identity has below a binary tree that represents the evolving time. In this setting a user who is given $\mathsf{sk}_{\mathsf{ID},0}$ can derive the secret keys for all the nodes in its binary subtree, that is for all successive time periods. In order to achieve forward-security, users are required to update their keys every time the period expires. More precisely a user computes $\mathsf{sk}_{\mathsf{ID},t+1} \xleftarrow{\$} \mathsf{KeyUpdate}(\mathsf{sk}_{\mathsf{ID},t})$ and deletes $\mathsf{sk}_{\mathsf{ID},t}$. The construction is based on the decisional weak ℓ-Bilinear Diffie Hellman Inversion Assumption (ℓ-wBDHI* for short) that was introduced by Boneh, Boyen and Goh in [3]. The ℓ-wBDHI* problem is defined in a bilinear group G of prime order p where $g \in G$ is a generator. Given $D = (g, h, g^{\alpha}, g^{\alpha^2}, ..., g^{\alpha^{\ell}})$, for random $\alpha \in \mathbb{Z}_p^*$ and $h \in G$, we say that an algorithm $\mathcal{A}$ has advantage ϵ in solving decisional ℓ-wBDHI* in G if

$$\left| \Pr[\mathcal{A}(D, e(g,h)^{\alpha^{\ell+1}}) = 0] - \Pr[\mathcal{A}(D, e(g,g)^{z}) = 0] \right| \leq \epsilon$$

where the probability is taken over the random choices of $\alpha, z \in \mathbb{Z}_p^*$ and $h \in G$.

The ℓ-wBDHI* assumption holds in a bilinear group G if, for any ℓ polynomial in k, any polynomially bounded adversary $\mathcal{A}$ has at most negligible advantage.

The scheme follows:

$\mathsf{Setup}(1^k, \ell)$**.** Let G and G_T be two groups of prime order p equipped with a bilinear map $e : G \times G \to G_T$. Let $g \in G$ be a generator. Pick a random $\alpha \xleftarrow{\$} \mathbb{Z}_p^*$ and set $g_1 = g^{\alpha}$. Then take random elements $g_2, u, v, h_1, \ldots, h_{\ell} \xleftarrow{\$} G$, compute $z = e(g_1, g_2)$ and select an hash function $H : \{0,1\}^* \to \mathbb{Z}_p^*$. The master public key is $MPK = (p, G, G_T, g, g_1, g_2, z, u, v, h_1, \ldots, h_{\ell}, H)$ and the master secret key is $MSK = g_2^{\alpha}$.

$\mathsf{KeyGen}(MSK, \mathsf{ID}, t)$**.** Let $\mathsf{sk}_{\mathsf{ID},w}$ be the key of the node w where w is a binary string of length at most ℓ. A key $\mathsf{sk}_{\mathsf{ID},t}$ is organized as a stack of node keys where $\mathsf{sk}_{\mathsf{ID},w^t}$ is on top.
A node key $\mathsf{sk}_{\mathsf{ID},w^t}$ is computed as follows. Let $w^t = w_1 \cdots w_k$ (with $0 \leq k \leq \ell$) be the binary string representing the node w^t. Pick a random $r \xleftarrow{\$} \mathbb{Z}_p^*$ and compute $d_0 = g_2^{\alpha}(uv^{H(\mathsf{ID})} \prod_{i=1}^{k} h_i^{f(w_i)})^r$, $d_1 = g^r$, $b_i = h_i^r$ for $i = k+1$ to ℓ. Since $0 \notin \mathbb{Z}_p^*$ $f : \{0,1\} \to \mathbb{Z}_p^*$ is a function that maps 0 and 1 to specific values of $\mathbb{Z}_p^*$ (e.g $f(0) = 1, f(1) = 2$). Thus we have $\mathsf{sk}_{\mathsf{ID},w^t} = (d_0, d_1, b_{k+1}, \ldots, b_{\ell})$. Finally $\mathsf{sk}_{\mathsf{ID},t}$ contains all the node keys of the stack that are needed to derive the keys of successive time periods. We notice that the stack will contain at most $O(\ell)$ node keys.

$\mathsf{KeyUpdate}(\mathsf{sk}_{\mathsf{ID},t})$**.** First pop the first key from the stack. If w^t is a leaf node, then $\mathsf{sk}_{\mathsf{ID},w^{t+1}}$ is the next key on the stack. Otherwise, if w^t is an internal node, compute $(\mathsf{sk}_{\mathsf{ID},w^t0}, \mathsf{sk}_{\mathsf{ID},w^t1})$ as described below and push $\mathsf{sk}_{\mathsf{ID},w^t1}$ and then $\mathsf{sk}_{\mathsf{ID},w^t0}$ onto the stack. In both cases the node key $\mathsf{sk}_{\mathsf{ID},w^t}$ is erased. Given the node key $\mathsf{sk}_{\mathsf{ID},w^t} = (d_0, d_1, b_{k+1}, \ldots, b_\ell)$, a key $\mathsf{sk}_{\mathsf{ID},w^tb}$ for $b \in \{0,1\}$ is obtained as follows. Pick a random $t \xleftarrow{\$} \mathbb{Z}_p^*$ and compute

$$d_0' = d_0(uv^{H(\mathsf{ID})} \prod_{i=1}^{k} h_i^{f(w_i)} h_{k+1}^{f(b)})^t b_{k+1}^{f(b)}, \;\; d_1' = d_1 g^t, \;\; b_i' = b_i h_i^t$$

for $i = k+2$ to ℓ. It is easy to notice that such key is correctly distributed for randomness $r + t$.

$\mathsf{Encrypt}(\mathit{MPK}, \mathsf{ID}, t, m)$. Let $w^t = w_1 \cdots w_k$ be the node of the tree associated with t. Pick a random $s \xleftarrow{\$} \mathbb{Z}_p^*$ and compute $C_0 = (uv^{H(\mathsf{ID})} \prod_{i=1}^{k} h_i^{f(w_i)})^s$, $C_1 = g^s$ and $C_2 = z^s m$. Finally output $C = (C_0, C_1, C_2)$.

$\mathsf{Decrypt}(\mathsf{sk}_{\mathsf{ID},t}, C)$. The message is recovered by computing $m = \frac{C_2 e(C_0,d_1)}{e(C_1,d_0)}$.

The security of the scheme follows from the following theorem whose proof is omitted for lack of space.

Theorem 2. *The scheme is fs-IND-ID-CPA secure if the decisional $(\ell+1)$-wBDHI* holds and H is modeled as a random oracle.*

Now we show how to convert the construction given above into a IND-CCA secure fs-IB-KEM using a very simple variant of Dent's transform [10]. Such a transform allows to convert a forward secure IBE satisfying very weak security requirements into an IND-CCA secure fs-IB-KEM. Specifically the underlying IBE is required to be only one-way forward secure.

Suppose $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{KeyUpdate}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a secure (in the weak sense mentioned above) fs-IBE scheme with a finite and efficiently sampleable message space $\mathcal{M}$. We assume that the $\mathsf{Encrypt}$ algorithm uses random values taken from a set R. We can write $\mathsf{Encrypt}$ as a deterministic algorithm $C \leftarrow \mathsf{Encrypt}(\mathit{MPK}, \mathsf{ID}, t, m; r)$ where $r \xleftarrow{\$} R$. The only difficulty in applying the method of Dent [10] is that we must re-encrypt the recovered message for integrity check. In the context of forward secure IBEs, this means one must know the time period and the identity under which the message was originally encrypted. In our setting (i.e. the specific application of onion routing) we overcome this difficulty as such information is available to routers.

We transform the fs-IBE scheme Π with a finite and efficiently sampleable message space $\mathcal{M}$ and maximum number of time periods T into a fs-IB-KEM scheme $\Pi' = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{KeyUpdate}, \mathsf{Encap}, \mathsf{Decap})$ using two hash functions:

$$H_1 : \{0,1\}^* \times \{0,1\}^* \times \mathcal{M} \to R \qquad \text{and} \qquad H_2 : \{0,1\}^* \to \{0,1\}^k .$$

6 Efficiency and Comparisons

In this section we compare the efficiency of our proposal with those of the other known solutions: the certificateless onion routing (CL-OR) protocol of Catalano *et al.* [7], the pairing-based onion routing (PB-OR) scheme of Kate *et al.* [21]

and the actual Tor protocol (we refer to the official specifications [13]). Basically, all these solutions differ only in the way the symmetric keys are established, so we decided to analytically compare the cost of building a circuit of length n from the perspective of both a user and an onion router. All the tests are carried on considering security parameters of 80 and 128 bits: as widely suggested in [31,32], the latter should be considered in order to gain an adequate long-term security level.

In what follows we briefly describe the operations involved during the building of a circuit in the considered protocols.

Tor. The Tor protocol incrementally builds the circuit using the telescoping technique and each new key is established using a Diffie-Hellman (DH) key-exchange [12]. Its specifications require that the user sends to each onion router the DH component encrypted using an RSA key associated to the router. It follows that: a user computes 1 RSA encryption and 2 exponentiations for each of the n routers; an onion router performs 1 RSA decryption and 2 exponentiations. Even if not required by Tor's specifications [13] we consider pre-computation on the fixed base for one of the two exponentiations of the DH key-exchange. For a security level of 80 bits we need a 1024-bits RSA modulus and a 1024-bits finite field for Diffie-Hellman. The specifications given in [13] suggest to use 65537 as fixed RSA exponent and to optimize DH with exponents of 360 bits and generator 2. In order to get a 128-bits security level, the sizes of the RSA modulus as well as of the DH finite-field have to be of 3072 bits. Tor's specifications [13] require a periodic update of the onion routers' keys.

PB-OR. We consider an implementation of the pairing-based onion routing protocol over a group of points of elliptic curves using the PBC library [22]. More specifically, following the indications of the authors, a type A (in the PBC nomenclature) curve is used in order to get fast pairing operation $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Kate *et al.* suggest that each user can pre-compute a pairing application for each onion router (as a function of some public parameter and of onion router's identity); such values have to be re-computed every time the KGC's keys change (e.g. every day). Therefore, in order to build a circuit of length n, a user has to compute n exponentiations in $\mathbb{G}$ and n exponentiations in $\mathbb{G}_T$: both the operations can be speed-up using pre-computation on the fixed base. On the other hand, each onion router has to compute one pairing but an optimized implementation can exploit a pre-computation on the pairing application that makes use of a fixed parameter (such functionality is offered by PBC library).

CL-OR. For the CL-OR protocol[2] of Catalano *et al.* we also consider an implementation over EC using the PBC library but, as suggested by the authors, using a type F curve in order to gain fast operations on smaller group elements. The user can pre-precompute some values that are function of the onion router's identities and public-keys. The on-line computation of the user requires

[2] In [7] there are two implementations available: we consider the fastest that makes use of the Strong Diffie-Hellman assumption in the Random Oracle model.

3 exponentiations on the working group: only 2 of them can be performed using pre-computation on the fixed bases. In such protocol, each onion router requires 2 exponentiations to compute the session-key: none of them can use such kind of pre-computation.

Our Protocol. For this comparison we consider our proposal of Section 5 implemented on a type A curve (like the PB-OR protocol). Observe that to compute $C_0 = (uv^{H(\mathsf{ID})} \prod_{i=1}^{k} h_i^{f(w_i)})^s$, many values can be entirely pre-computed off-line by the user: the values $uv^{H(\mathsf{ID})}$ as well as the values $h_i^{f(w_i)}$. The remaining notable tasks for the user, for each onion router in the circuit, are: two exponentiations over $\mathbb{G}$ (one for C_0 and one for $C_1 = g^s$) and one exponentiation in $\mathbb{G}_T$ to compute $C_2 = z^s m$. Notice that the latter two exponentiations can be optimized with pre-computation on the fixed bases g, z. Each onion router involved in the circuit establishment has to compute 2 pairings for decryption (but with partial pre-computation as in PB-OR) as well as a new encryption to fulfill the integrity check required by Dent's transformation.

In a fully operational implementation of an onion routing network, the key establishment phase involves the use of other minor tools: a symmetric encryption scheme (e.g. AES) to protect the passing messages using the negotiated session keys and fixed TLS channels among the connected onion routers. For sake of simplicity we ignore the computational load related to such operations since they are used by all the protocols considered in our comparisons. Moreover, in the case of AES, its time complexity is negligible if compared with the other involved cryptographic tools.

As a first step, all the operations were implemented using the PBC library (version 0.4.18) on a 2.4GHz Intel Core 2 Duo workstation running Mac OS X 10.5.6[3].

As one can see from Table 1, from a purely computational perspective our solution is not faster than previous protocols but its computational costs are definitely practical. In these comparisons it is worth noting that the Tor circuit construction is an interactive protocol that requires a quadratic number of exchanged messages. Therefore if we consider the natural network latency we obtain that, even for the shortest possible circuit (i.e. 3 nodes), our protocol is faster than Tor in constructing an entire new circuit. In fact, assuming a network latency of 50 ms, Tor requires 627 ms to complete the circuit construction while our protocol needs only 370 ms.

A LOOK AT INTERACTION. We stress that the main contribution of our work is that the resulting OR protocol is totally non-interactive solving a problem that is yet unsolved in currently known solutions. Indeed, all the other onion routing protocols require interaction in some phase of the protocol. Tor is clearly interactive in the circuit construction phase due to the use of telescoping. In PB-OR the routers have to obtain new private keys from the KGC every time the key validity period expires (a heavy workload for the KGC!). On the other hand,

[3] The computational costs of each operation are ommitted for lack of space. They are available in the full version of this work.

Table 1. Total times for building a circuit of n routers and protocol features

Times and features	Tor		PB-OR		CL-OR		our protocol	
	user	OR	user	OR	user	OR	user	OR
Time for 80 bits security (ms)	$2.3n$	6.9	$1.1n$	3.9	$2.1n$	3.4	$7.8n$	15.6
Time for 128 bits security (ms)	$16.5n$	93.3	$9.3n$	57.3	$5.1n$	8.2	$63.4n$	178.0
Number of exchanged messages	$n(n+1)$		$2n$		$2n$		$2n$	
IND-CPA security level	×		✓		✓		✓	
IND-CCA security level	×		×		✓		✓	
Non-inter. circuit construction	×		✓		✓		✓	
Non-inter. key-update by OR	×		×		✓		✓	
Non-inter. by user after OR key-update	×		✓		×		✓	
Absence of key-escrow by KGC	✓		×		✓		×	

in CL-OR the routers can generate the updated keys by themselves but the users have to obtain such new keys (e.g. by querying a directory server) every time they are changed.

We stress that in our protocol onion routers can update their keys without interacting with any party, and this process is transparent for the users who keep using always the same public keys (i.e. routers' identities). It is interesting to note that the non-interactive nature of our key update allows to reduce the security gap between eventual and immediate forward-security. We can indeed arbitrarily reduce the refresh period of routers' keys without any further network overhead: this is not true for PB-OR and CL-OR. Finally, we observe that the slightly higher computational cost of our solution is due only to the fact the best fs-IB-KEM we can achieve has to perform pairings computation. Although this is currently a limitation, we believe that the purely non-interactive nature of our protocol, more than compensate for the slight increase in computational cost.

References

1. Al-Riyami, S., Paterson, K.: Certificateless public key cryptography. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 452–473. Springer, Heidelberg (2003)
2. Anderson, R.: Two remarks on public key cryptology. In: ACM-CCS 1997 (1997) (invited lecture), `http://www.cl.cam.ac.uk/ftp/users/rja14/forwardsecure.pdf`
3. Boneh, D., Boyen, X., Goh, E.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
4. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–615. Springer, Heidelberg (2001)
5. Camenisch, J., Lysyanskaya, A.: A Formal Treatment of Onion Routing. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 169–187. Springer, Heidelberg (2005)

6. Canetti, R., Halevi, S., Katz, J.: A Forward-Secure Public Key Encryption Scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)
7. Catalano, D., Fiore, D., Gennaro, R.: Certificateless Onion Routing. In: Proc. of the 16th ACM Conference on Computer and Comm. Security (CCS 2009), pp. 151–160. ACM Press, New York (2009)
8. Chaum, D.: Untraceable Electronic Mail, return address and digital pseudonyms. Communications of the ACM 24(2), 84–88 (1981)
9. Dai, W.: PipeNet 1.1, http://www.weidai.com/pipenet.txt
10. Dent, A.W.: A designer's guide to kEMs. In: Paterson, K.G. (ed.) Cryptography and Coding 2003. LNCS, vol. 2898, pp. 133–151. Springer, Heidelberg (2003)
11. Danezis, G., Goldberg, I.: Sphinx: A Compact and Provably Secure Mix Format. In: IEEE Symposium on Security and Privacy 2009, pp. 269–282 (2009)
12. Diffie, W., Hellman, M.: New Directions in Cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)
13. Dingledin, R., Mathewson, N.: Tor Protocol Specification (2008), http://www.torproject.org/svn/trunk/doc/spec/tor-spec.txt
14. Dingledin, R., Mathewson, N., Syverson, P.: Tor: The Second-Generation Onion Router. In: Proc. of the 13th USENIX Security Symposium, pp. 303–320 (2004)
15. Freedman, M., Morris, R.: Tarzan: A Peer-to-Peer Anonymizing Networ Layer. In: Proc. of 9th ACM Conference on Computer and Comm. Security (CCS 2002), pp. 193–206 (2002)
16. Goldberg, I.: On the Security of the Tor Authentication Protocol. In: Danezis, G., Golle, P. (eds.) PET 2006. LNCS, vol. 4258, pp. 316–331. Springer, Heidelberg (2006)
17. Goldschlag, D., Reed, M., Syverson, P.: Hiding Routing Informations. In: Anderson, R. (ed.) IH 1996. LNCS, vol. 1174, pp. 137–150. Springer, Heidelberg (1996)
18. Goldschlag, D., Reed, M., Syverson, P.: Onion Routing for Anonymous and Private Internet Connections. Communications of the ACM 42(2), 84–88 (1999)
19. Kate, A., Goldberg, I.: Using Sphinx to Improve Onion Routing Circuit Construction. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 359–366. Springer, Heidelberg (2010)
20. Kate, A., Zaverucha, G., Goldberg, I.: Pairing-Based Onion Routing. In: Borisov, N., Golle, P. (eds.) PET 2007. LNCS, vol. 4776, pp. 95–112. Springer, Heidelberg (2007)
21. Kate, A., Zaverucha, G., Goldberg, I.: Pairing-Based Onion Routing with Improved Forward Secrecy. ACM Transactions on Information and System Security (2009)
22. Lynn, B.: PBC: The Pairing-Based Crypto Library, http://crypto.stanford.edu/pbc
23. Möller, B.: Provably Secure Public-Key Encryptionfor Length-Preserving Chaumian Mixes. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 244–262. Springer, Heidelberg (2003)
24. Øverlier, L., Syverson, P.F.: Improving Efficiency and Simplicity of Tor Circuit Establishment and Hidden Services. In: Borisov, N., Golle, P. (eds.) PET 2007. LNCS, vol. 4776, pp. 134–152. Springer, Heidelberg (2007)
25. Reed, M., Syverson, P., Goldschlag, D.: Anonymous Connections and Onion Routing. IEEE Journal on Selected Ares in Communications 16(4), 482–494
26. Renhard, M., Plattner, B.: Introducing MorphMix: Peer-toPeer based Anonymous Internet Usage with Collusion Detection. In: The Workshop on Privacy in the Electronic Society (WPES 2002), pp. 91–102. ACM, New York (2002)

27. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
28. Shoup, V., Gennaro, R.: Securing threshold cryptosystems against chosen ciphertext attack. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 1–16. Springer, Heidelberg (1998)
29. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairing. In: Symposium on Cryptography and Information Security, Okinawa, Japan (2000)
30. Yao, D., Fazio, N., Dodis, Y., Lysyanskaya, A.: ID-Based Encryption for Complex Hierarchies with Applications to Forward Security and Broadcast Encryption. In: Proc. of the ACM Conference on Computer and Comm. Security, CCS 2004 (2004)
31. NIST Recommendations for Key Management Part 1: General NIST Special Publication 800-57 (August 2005), `http://csrc.nist.gov/publications/nistpubs/800-57/SP800-57-Part1.pdf`
32. ECRYPT Yearly Report on Algorithms and Key Sizes (2007-2008) (July 2008), `http://www.ecrypt.eu.org/ecrypt1/documents/D.SPA.28-1.1.pdf`