Certificateless Onion Routing

Dario Catalano
Dipartimento di Matematica e Informatica
Università di Catania - Italy
catalano@dmi.unict.it

Dario Fiore
Dipartimento di Matematica e Informatica
Università di Catania - Italy
fiore@dmi.unict.it

Rosario Gennaro
IBM T.J. Watson Research Center
Hawthorne, New York 10532
rosario@us.ibm.com

ABSTRACT

Onion routing protocols allow users to establish anonymous channels to preserve their privacy over a public network. Several protocols implementing this primitive have been proposed in recent years, and TOR, a real-life implementation, provides an onion routing service to thousands of users over the internet.

This paper presents *Certificateless Onion Routing* a new approach to the problem. Starting from the identity based solution (PB-OR) of Kate *et al.* [23], we adopt the certificateless setting introduced by Al-Riyami and Paterson [2]. Such a setting is particularly well suited in practice as it retains the good aspects of identity based cryptography (no PKI is required) and traditional public key cryptography (there is no key escrow). Next, we present a novel *certificateless anonymous key-agreement* (KA) protocol and we show how to turn it into a very efficient (and provably secure!) certificateless onion routing protocol. When compared with Tor and PB-OR, our protocol offers better performances, especially when current security levels (i.e. 128 bits) are considered. In particular, our scheme significantly improves the computational costs required from each router. In this sense our solution is up to 7 times faster than PB-OR and up to 11 times faster than Tor.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*

General Terms

Security

1. INTRODUCTION

Imagine that some user wishes to anonymously communicate with other parties using a public network (like the internet). Here by anonymously we mean that the user wishes to hide her identity and also her network information (e.g. her network address). Achieving this level of anonymity is of primary 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.). The notion of anonymous communication was first introduced by Chaum in 1981 [10], with a solution to achieve an *anonymous channel*. Very informally, the basic idea there was to "wrap" messages in several layers of encryptions and send them through a sequence of nodes. Each node, when receiving a ciphertext, peels off a layer of encryption and sends the resulting value to the next node. Anonymity derives from the fact that the order in which the nodes are selected is random and that each node should learn nothing more than its two adjacent nodes in the sequence.

Goldschlag *et al.* introduced in [20] the notion of *Onion Routing* that extends Chaum's idea as follows. An onion routing protocol is defined by a service provider, users and a set of nodes (called *onion routers*). A user that wishes to send a message selects a random set of nodes, wraps the message with several layers of encryption, one for each node, and sends it through these intermediate nodes (a *circuit*). When a node receives a message, it decrypts it and sends the resulting value to the next node. Because of their layered composition such wrapped messages are called *onions*.

This very simple and elegant idea led to several other constructions and implementations (e.g. [20, 11, 29, 21, 18, 30, 15]). Perhaps the most famous of such solutions is the so-called Onion Routing Project, recently replaced by Tor [15] (the second generation onion router). The goal of Tor is to provide anonymity to users over the Internet. At the moment it counts, roughly, 1000 onion routers and hundreds of thousands of users over the world.

The most important component of an onion routing protocol is how a user builds a circuit, namely how a secure channel is established through each onion router. In the construction given in [20] the user sends to each onion router a message encrypted with its public key, containing a random symmetric key (used to encrypt the corresponding onion layer) and the name of the next node in the circuit. An obvious drawback of this solution is that it cannot tolerate corruptions. Indeed, imagine an adversary that corrupts some router O in order to get its private key. Using this information the adversary can then decrypt all the ciphertexts received by O thus obtaining *all* the session keys (past and present) of the router. We would like to have a protocol with a *forward secrecy* property, in which a router's corruption does not reveal anything about communication prior to the corruption.

A simple way to achieve forward secrecy is to frequently change the keys, so to minimize the period in which the attack can be successfully done. However such a solution can be very complicated in practice as it forces users to repeatedly obtain new keys for the routers; similarly routers have to generate such keys and also corresponding valid certificates must be issued.

Dingledine *et al.* [15] proposed for Tor a solution which relies on using the routers' public keys only to establish a temporary session key via an (interactive) Diffie-Hellman [13] key agreement. The Tor Authentication Protocol (TAP) uses a technique called *telescoping* to construct the circuit, and was formally proven secure by Goldberg [19]. Although Øverlier and Sylverson [27] further improved the efficiency of telescoping, the main issue of this technique is the bandwidth cost. Indeed, building a circuit of length n requires the exchange of $O(n^2)$ (symmetrically encrypted) messages.

In 2007 Kate *et al.* [22, 23] proposed a new approach to the problem, that they called *pairing-based onion routing* (PB-OR). In their work they exploited the features of two identity based schemes: the one of Boneh and Franklin [4] and the protocol of Sakai *et al.* [33].

Identity-based cryptography was introduced by Adi Shamir in [31] to simplify certificate management in traditional public key cryptography. The basic idea of this notion is that parties use their identities as public keys. In this setting each party receives his secret key from a trusted Key Generation Center, whose public parameters are publicly known.

The result of Kate *et al.* [23] is an onion routing protocol in the identity-based setting where the service provider acts as a Key Generation Center for routers' private keys. This approach has several advantages: the circuit construction is non-interactive and requires only $O(n)$ messages to be exchanged. Moreover certificates management (and verification) can be avoided. The authors also showed that the performance of building a circuit in PB-OR is better than in Tor. On the negative side, to achieve forward secrecy Kate *et al.* require to frequently change the keys of both the KGC and the routers. Although KGC's keys can have a long validity period, the KGC is involved in the expensive and interactive process of generating the private keys of all routers (which occurs very frequently, e.g. every hour as suggested in [23]).

Our contribution In our work we propose a new onion routing protocol in the *certificateless encryption* setting.

Certificateless encryption is an hybrid setting that lies between public key and identity-based cryptography. It was first introduced by Al-Riyami and Paterson in [2] (see [12] for a nice survey on the subject).

In the certificateless setting, as in the identity-based one, each user has a string ID, representing his identity, and a matching secret key produced by a KGC. Furthermore each user also has a public/secret key pair, as in the traditional public key model. Certificateless encryption gets the advantages of both identity-based and traditional public key cryptography (while avoiding their drawbacks). In particular, (i) the KGC cannot decrypt ciphertexts of users and (ii) public keys do not need to be certified. The only requirements is that the public parameters used by the KGC must be trusted by all the parties.

We construct our protocol in two steps. First, we define the notion of *certificateless anonymous key-agreement* (KA) and propose two constructions. The first one is very efficient and it is proven secure under the Strong-Diffie-Hellman Assumption [1][1]. The second solution is slightly less efficient, but it can be proved secure under the standard Computational Diffie-Hellman Assumption. Roughly speaking, an anonymous KA protocol allows a user (e.g. Alice) to establish a session key with another user (e.g. Bob) in such a way that Alice authenticates Bob, without revealing her identity.

Next, we present a certificateless onion routing protocol and show how to use our certificateless anonymous KA protocol for building the circuit. In our construction a user chooses a set of onion routers and runs the KA protocol to (non-interactively) establish a session key with each of them. These session keys are later used to form an onion in the usual way.

In order to achieve forward secrecy we also follow the approach of periodically changing keys. More precisely we require *only* onion routers to frequently update their keys but, unlike all previously known constructions, our scheme allows for a very simple and efficient update process. This is because in our protocol routers can generate keys on their own *without interacting with the KGC*. Moreover such keys *do not* need to be certified. Thus no additional interaction with trusted entities (either KGC or CA) is needed, with a significant reduction on their workload.

We compare the efficiency of our protocol with that of PB-OR [23] and Tor [14] and show that we achieve better performances, especially when high security levels (i.e. 128 bits) are considered. An additional important advantage of our protocol, with respect to PB-OR, is that we do not need expensive pairings computations. In particular we significantly improve (see section 4.2 for details) on the cost of running the protocol for each onion router. We stress that reducing the computational cost required from each onion router is very important in practice as the routers are required to perform much more operations than the user.

Other related work We refer the reader to the work in [26, 5] for formal security definitions for the problem of onion routing.

2. PRELIMINARIES

Let $\mathbb{N}$ the set of natural numbers. We denote with $\ell \in \mathbb{N}$ the security parameter. The partecipants to our protocols are modeled as probabilistic Turing machines whose running time is bounded by some polynomial in ℓ. If S is a (finite) set, we denote with $s \xleftarrow{\$} S$ the process of selecting an element uniformly at random from S.

Informally we say that a function is *negligible* if it vanishes faster than the inverse of any polynomial.

2.1 Onion Routing

An onion routing protocol is characterized by a service provider, a set of onion routers and users. The goal of users is to obtain anonymous access to the network by sending their traffic through a circuit of (randomly chosen) onion routers. A protocol is typically defined by the following phases:

Setup and Key Generation In this phase the service provider sets up some public parameters for the system and

[1]We remark that in recent papers the name Strong Diffie-Hellman was used to denote a different conjecture defined over bilinear groups [3]. In this paper, we refer to the original terminology from [1]

each user generates a pair of keys. Public keys need to be certified: this is typically done by the service provider. This phase can be slightly different if we are in an identity-based setting, such as the protocol of Kate *et al.* [23], or in a certificateless setting (as in our case).

Circuit construction The main goal of an onion routing protocol is to allow users to build a circuit of onion routers. This is typically done by establishing with each router a random session key. Once such a circuit is established, it can be exploited by users to route their traffic through it. In particular the user sends a message with several layers of encryption (one for each onion router) to the first router in the circuit. Then each onion router OR_i "peels off" a layer of encryption and obtains: (1) the name of the next router in the circuit, OR_{i+1} and (2) another ciphertext which is then forwarded to OR_{i+1}.

The ways in which a circuit is constructed can be different, but the common idea is that the onion routers are randomly chosen according to some strategy and the user anonymously establishes with each of them a session key that is used to encrypt the messages. Informally speaking, users achieve anonymity because each onion router learns only the identities of the two nodes adjacent to him in the circuit, or that he is the last node. Even the first node cannot recognize if it is receiving a message from a user or from an onion router.

A more formal and detailed definition of security for onion routing protocols is given in the next section.

2.1.1 *Security of onion routing*

Camenisch and Lysyanskaya [5] defined security for onion routing protocols in the universally composable (UC) framework and gave also a generic construction from CCA2 encryption *with tags* and pseudorandom permutations. Security in the UC framework is very strong as it automatically considers all possible attacks. Unfortunately, however, meeting such a security requirement often leads to inefficient schemes. Since we are primarily interested into practical solutions we follow the approach proposed by Kate *et al.* [23]. In particular we define security of onion routing protocols using the same properties stated in [23] and we replace *session key secrecy* with a stronger property: *onion-security.*

Cryptographic Unlinkability Informally speaking, this property states that it should be infeasible for an attacker to recognize a link between the sender and the receiver. Here "cryptographic" means that network-level attacks are not considered. Kate *et al.* gave a formal definition of this property and they also proved that it is implied by the IND-CPA security of the symmetric encryption scheme used in the protocol to form the onions. For lack of space we defer the interested reader to [23] for more details.

Integrity and Correctness *Correctness* guarantees that if all the parties correctly execute the protocol, then the recipient receives the message contained in the original onion prepared by the sender and forwarded through the circuit. In addition, an onion routing protocol has *integrity* if it is possible to recognize those onions that are longer than a pre-specified upper-bound.

Onion-security Informally, a protocol has *onion-security* if the session keys established between users and onion routers remain "secure" against an attacker that controls all but one honest node in the network. More formally we define below a security game between a PPT adversary $\mathcal{A}$ and a Challenger.

At the beginning the Challenger generates the public parameters of the system and gives them in input to the adversary. Then $\mathcal{A}$ is allowed to corrupt parties, namely it learns their secret keys and controls their actions. At some point the adversary chooses a circuit and a specific (honest) node O in it. The Challenger gives to $\mathcal{A}$ either the real session key between an anonymous sender and O or a random one. Then the adversary can continue to perform its actions without corrupting O and, at the end of the game, it must recognize which session key it received.

An onion routing protocol has onion-security if any PPT adversary $\mathcal{A}$ has at most (1/2 + negligible) probability of winning the above game.

A formal definition of this security game is presented in the next section. It follows the Canetti-Krawczyk (CK) model [7, 8] for key-agreement protocols, adapted to the certificateless model.

Circuit Position Secrecy When a protocol satisfies this property, an onion router cannot learn its position in the circuit. This means that exchanged messages should not reveal this information. In general it is not easy to achieve this property: if onions are constructed encrypting a message with several keys (e.g. C=$E_{K_1}(E_{K_2}(\ldots E_{K_n}(m)\ldots))$ then, as showed by Camenisch and Lysyanskaya in [5], the ciphertext grows proportionally to the number of times it is encrypted. This is because in randomized encryption schemes ciphertexts are longer then the messages. However some solutions for this issue are given in [5, 23].

3. CERTIFICATELESS ANONYMOUS KEY AGREEMENT

In this section we present the notion of *non-interactive one-way certificateless anonymous key agreement* (KA) that is derived from that of *pairing-based key agreement with users anonymity* presented by Kate *et al.* in [23]. While the protocol given in [23] is in the identity-based setting, ours is in an hybrid setting called *certificateless*[2]. In such a setting a user is defined by a string ID representing its identity and receives a partial secret key d_{ID} associated to ID from a trusted entity, called the Key Generation Center (KGC). From d_{ID} the user can later derive a pair of keys which can be used as in the usual public key way.

The main advantage of certificateless encryption is that it retains the good aspects of identity-based encryption and traditional public key encryption while avoiding their drawbacks: the KGC cannot decrypt ciphertexts of users and public keys do not need to be certified. Only the public parameters of the KGC must be trusted by all parties.

A one-way anonymous KA protocol allows a user (e.g. Alice) to establish a session key with another user (e.g. Bob) in such a way that Alice authenticates Bob, without leaking her identity. It is also possible to consider a *two-way* protocol in which both the parties are anonymous, but this is out of the interest of this work. We consider a non-interactive version of such protocols where there is a single message sent by the (anonymous) sender to the (non-anonymous) receiver. In what follows we use the term "certificateless anonymous key agreement" to refer to this non-interactive one-way version.

[2]In [12] Dent gives a nice survey presentation of this paradigm (restricted to encryption schemes) and analyzes several definitions

The protocol has a setup phase in which the KGC generates the public parameters of the system and issues partial secret keys to users associated with their public identities, e.g. the KGC generates a partial secret key d_{ID} associated to ID. Then each user can generate on its own a pair of keys as a function of the received partial secret key. We assume that the system has a list in which each user can publish its identity and the associated public key. Such a list can also be used by users to know the set of the available parties in the protocol.

Once the setup phase is over, a user U can choose a random pseudonym P_U and run the protocol with another party ID chosen from this public list. In particular the anonymous user computes the session key as a function of its pseudonym and the other party's public key and then sends P_U to ID. The recipient can later use its secret key and the received pseudonym to derive the same session key computed by U.

Following Kate *et al.* [23] we require that an anonymous certificateless KA protocol must satisfy the following properties:

Unconditional Anonymity: It should be impossible to learn the identity of an anonymous party running the protocol.

Session Key secrecy. Kate *et al.* required in [23] that an attacker should not be able to recover session keys of uncorrupted parties. We adopt a much stronger (and standard) notion of security: we require that any PPT attacker $\mathcal{A}$ should not be able to distinguish a random session key from a real one for an adaptively chosen recipient. More formally we define two games, Type I and Type II, and we require that $\mathcal{A}$ should have at most *negligible* advantage in these games. We point out that the third property stated in [23], *no impersonation*, is captured by our session key secrecy.

TYPE I SECURITY

Setup The Challenger generates (MPK, MSK), a pair of keys for the KGC and gives MPK to $\mathcal{A}$.

Phase 1 In this phase the adversary is allowed to adaptively perform the following actions:

- asking partial private keys of users. On input ID the challenger extracts d_{ID} using MSK and provides d_{ID} to $\mathcal{A}$.
- issuing private key extraction queries. When the challenger receives such query with input ID, it extracts d_{ID} (if it was not generated before) and then computes a pair of keys (pk_{ID}, sk_{ID}) using d_{ID}. It gives sk_{ID} as response to $\mathcal{A}$.
- requesting public keys of users. In this case the challenger proceeds as in the previous case and returns in output pk_{ID}.
- asking session keys. After receiving an identity ID and a pseudonym P from the adversary, the challenger outputs a valid session key for a protocol session between P and ID.
- replacing the public key of a user. The adversary provides an identity ID and a public key pk'_{ID} meaning that it wants to replace a previously generated public key with pk'_{ID}. Note that the adversary will not be later allowed to ask the private key related to pk'_{ID} nor a session key of a session involving ID.

Challenge At some point $\mathcal{A}$ outputs a target identity ID^*. The challenger picks a random pseudonym P and defines K_0 as the real session key between P and ID^* while K_1 is taken at random. Then it picks a random bit $b \in \{0, 1\}$ and gives K_b to the adversary. Note that $\mathcal{A}$ is not allowed to choose an identity ID^* for which a private key was extracted in the previous phase.

Phase 2 This phase is executed as Phase 1 except that the adversary cannot query ID^* to the private key extraction oracle.

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

We define the advantage of an adversary $\mathcal{A}$ in the game above as $\mathbf{Adv}^I_{\mathcal{A}} = |\Pr[b = b'] - 1/2|$.

TYPE II SECURITY. Type II security models the fact that a KGC should not be able to break the security of the protocol when it is passive, namely in the case when it does not replace public keys trying to *actively* impersonate users.

To formalize Type II security we define a game which consists of the same phases as that of Type I security with the following different rules:

- $\mathcal{A}$ receives in input the master secret key MSK;
- $\mathcal{A}$ cannot query the partial private key extraction oracle (observe that it can generate such keys on its own);
- $\mathcal{A}$ cannot replace public keys.

We define the advantage of $\mathcal{A}$ in this game as $\mathbf{Adv}^{II}_{\mathcal{A}} = |\Pr[b = b'] - 1/2|$.

We point out that this kind of definition of security (Type I and Type II) follows standard cryptographic definitions for certificateless encryption schemes (see Dent's survey [12] for more details).

3.1 Our certificateless anonymous KA protocol

In this section we present our protocol for certificateless anonymous key-agreement whose security relies on the Strong-DH Assumption [1] in the random oracle model. The protocol uses techniques similar to that used by Fiore and Gennaro in [17] to construct an identity-based key-agreement protocol.

Protocol setup. The KGC chooses a group $\mathbb{G}$ of prime order q (where q is ℓ-bits long), a random generator $g \in \mathbb{G}$ and two hash functions $H_1 : \{0, 1\}^* \to \mathbb{Z}_q$ and $H_2 : \mathbb{Z}_q \times \mathbb{Z}_q \to \{0, 1\}^\ell$. Then it picks a random $x \xleftarrow{\$} \mathbb{Z}_q$ and sets $y = g^x$. Finally the KGC outputs the public parameters $MPK = (q, \mathbb{G}, g, y, H_1, H_2)$ and keeps the master secret key $MSK = x$ for itself.

Partial Secret Key Extraction. A user with identity ID receives, as its partial secret key, a Schnorr's signature [34] of the message $m = ID$ under public key y. More specifically, the KGC after verifying the user's identity, creates the associated secret key as follows. First it picks a random $k \xleftarrow{\$} \mathbb{Z}_q$ and sets $r = g^k$. Then it uses the master secret key x to compute $s = k + H_1(ID, r)x$. The partial secret key returned to the user is $d_{ID} = (r, s)$.

User's key generation. Once a user ID has obtained a partial secret key d_{ID} from the KGC, it can generate its own pair of keys. It picks a random $t \xleftarrow{\$} \mathbb{Z}_q$ and sets $u = g^t$. Then it defines $\mathsf{pk}_{ID} = (r, u)$ and $\mathsf{sk}_{ID} = (s, t)$.

A protocol session. When a user U wants to establish a session key with Bob, it proceeds as follows. U selects a random $w \xleftarrow{\$} \mathbb{Z}_q$ and defines $P_U = g^w$ as its pseudonym. Then it searches Bob in the public list of the system and gets Bob's identity B and public key $\mathsf{pk}_B = (r_B, u_B)$. U computes the session key $K = H_2(z_1, z_2)$ where $z_1 = (r_B y^{H_1(B,r_B)})^w$ and $z_2 = u_B^w$ and sends its pseudonym P_U to Bob. Upon receipt of P_U Bob can recover the same session key by computing $z_1 = P_U^{s_B}$ and $z_2 = P_U^{t_B}$. It is easy to see that both the parties are computing the same values $z_1 = g^{ws_B}$ and $z_2 = g^{wt_B}$.

Theorem 1. *The protocol given above is a secure certificateless anonymous key-agreement protocol under the Strong-DH Assumption if H_1 and H_2 are modeled as random oracles.*

3.2 Proof of security

We prove the security of our anonymous key agreement protocol under the Strong-DH Assumption (and in a slightly slower variant under the weaker CDH Assumptions). These computational assumptions are recalled in the Appendix.

The proof uses a typical reduction argument. In the following sections we show how to reduce the existence of an attacker that breaks the security of the protocol into an algorithm (i.e. the simulator) that is able to break the Strong-DH Assumption with non-negligible probability.

The proof of Theorem 1 is splitted into two lemmas: one for Type I security and the other one for Type II security. For lack of space some proofs are deferred to the final version.

3.2.1 Exponential Challenge-Response Signatures

Before proving Type I security, we introduce the notion of *Challenge-Response Signatures* which are instrumental in obtaining a reduction for this case.

Exponential Challenge-Response (XCR) Signatures were introduced by Krawczyk in [24] as a building block for the proof of the HMQV key-exchange protocol. Variants of such signatures were also used in [17] to prove the security of their identity-based key-agreement protocol. Roughly speaking XCR signatures consist of an interactive signing process where the recipient of a signature gives a *challenge* to the signer and the latter generates the signature on a message with respect to this challenge. Only who creates the challenge will be able to verify the correctness of the signature. We briefly recall the XCR = $(XKG, XSIG, XVER)$ signature scheme by Krawczyk [24] which is based on the Schnorr's signature scheme.

XCR signature scheme. The key generation algorithm $XKG(1^\ell)$ takes as input the security parameter ℓ, chooses a ℓ-bit prime q and a group $\mathbb{G}$ of order q. Then it picks a random $x \xleftarrow{\$} \mathbb{Z}_q$ and outputs the verification key $y = g^x$ and the secret key x.

A user wishing to receive a signature, first generates a challenge value $T = g^w$ for random $w \xleftarrow{\$} \mathbb{Z}_q$ and gives T to the signer. To produce a signature on a message m the signer runs $XSIG(x, m, T)$ which chooses random $k \xleftarrow{\$} \mathbb{Z}_q$, sets $r = g^k$ and $s = k + H(m, r)x$ where $H : \{0, 1\}^* \to \mathbb{Z}_q$ is an hash function. Finally it computes $z = T^s$ and outputs the pair (r, z) as the signature for m.

The recipient can verify the signature (r, z) with respect to a message m, public key y, and challenge $(T = g^w)$ for which it knows w by checking if $z \overset{?}{=} (ry^{H(m,r)})^w$ (this is the verification algorithm $XVER$).

An interesting property of this scheme which is important in our work is that a signer can pre-compute (or receive) a *signing-token* $(r = g^k, s = k + H(m, r)x)$ for a message m and then is able to generate signatures on that message for every challenge T (i.e. to compute the signature it outputs $(r, z = T^s)$).

Definition 1 (Security of XCR). *The* XCR *signature scheme is said to be secure if any PPT forger algorithm $\mathcal{F}$ has at most negligible probability of winning the game below.*

Setup The Challenger runs the key generation algorithm $(y, x) \leftarrow XKG(1^\ell)$, generates a challenge $T = g^w$ for random $w \xleftarrow{\$} \mathbb{Z}_q$ and runs the forger $\mathcal{F}$ on input (y, T).

Signing queries $\mathcal{F}$ is provided access to a *token-signing oracle* $TokSig(x, \cdot)$ that, given in input a message m, outputs a *signing-token* (r, s) for m.

Forgery The forger wins the game if it outputs a tuple (m^*, r^*, z^*) such that: (i) (r^*, z^*) is a valid signature with respect to the message m^* and the challenge T and (ii) m^* was not queried to the oracle $TokSig(x, \cdot)$.

We point out that this definition of security is slightly different from the one given in [24] and follows the same modifications proposed in [17]. More precisely we provide the forger with access to the more generic oracle $TokSig(x, \cdot)$ instead of an oracle that outputs signatures when queried on a message-challenge pair.

Theorem 2. *The* XCR *signature scheme is secure according to Definition 1 under the CDH Assumption if H is modeled as a random oracle.*

3.2.2 Type I Security

Now we can prove that our protocol achieves Type I security. We point out that, although XCR is secure under the CDH assumption, our reduction holds under the Strong-DH Assumption (as we need access to a DH oracle in the reduction).

Lemma 1. *Our certificateless anonymous KA protocol is Type I-secure if the* XCR *signature scheme is secure and H_2 is modeled as a random oracle.*

Proof. Let $\mathcal{A}$ be an adversary that has non-negligible advantage ϵ into breaking Type I security of the protocol. Then we show how to build a simulator S that succeeds into breaking the security of XCR with non-negligible probability.

The simulator acts as a forger for XCR and at the same time has to simulate the environment for the execution of $\mathcal{A}$ in each phase.

Setup S receives in input a tuple $(q, g, \mathbb{G}, y, T)$ and is also given access to the random oracle H and a token signing oracle $TokSig$. S runs $\mathcal{A}$ on input $MPK = (q, g, \mathbb{G}, y, H_1, H_2)$ where $(q, g, \mathbb{G}, y)$ is S's input, $H_1 = H$ and H_2 is a random oracle controlled by S as described below. When the simulator receives in input a query $H_2(z_1, z_2)$ it picks a random string $R \xleftarrow{\$} \{0, 1\}^\ell$, outputs R and stores $\langle z_1, z_2, R\rangle$ into a table $\overline{H_2}$. At the beginning of the game S guesses the user that the adversary will ask in the challenge phase. Assume it is Bob. If n is an upper bound to the number of parties in the protocol, where n is polynomial in the security parameter, then S's guess is right with probability $1/n$.

Phase 1 In this phase S has to simulate all the oracles provided to $\mathcal{A}$.

- To respond to a partial private key extraction query for user ID, S queries $TokSig$ on input ID and obtains (r, s). It outputs $d_{ID} = (r, s)$ and stores $\langle ID, r, s\rangle$ in $\mathsf{PartialKeyList}$.
- When the simulator receives a private key extraction query for user ID it proceeds as follows. If the queried user is Bob S returns "Abort" and terminates the simulation. Otherwise it searches in $\mathsf{PartialKeyList}$ to see if it contains a tuple $\langle ID, r, s\rangle$. If such a tuple does not exist it proceeds as in the step before to obtain it. Then it picks a random $t \xleftarrow{\$} \mathbb{Z}_q$, sets $u = g^t$ and stores $\langle ID, r, u\rangle$ in $\mathsf{PubKeyList}$ and $\langle ID, s, t\rangle$ in $\mathsf{PrivKeyList}$. Finally it outputs $sk_{ID} = (s, t)$.
- When the simulator receives a public key extraction query for user ID it proceeds as in the previous step with the following difference. If the queried user is Bob, instead of aborting the simulation, it generates random $r = g^k, u = g^t$ by picking $k, t \xleftarrow{\$} \mathbb{Z}_q$ and returns in output $pk_{ID} = (r, u)$.
- When $\mathcal{A}$ submits a public key replace query $pk'_{ID} = (r', u')$, the simulator searches $\mathsf{PubKeyList}$ for a tuple $\langle ID, r, u\rangle$, replaces (r, u) with (r', u') and marks the tuple as "updated".
- On receiving a session key query (P, ID) where $P = g^w$ is a random pseudonym, S searches $\mathsf{PubKeyList}$ for a tuple $\langle ID, r, u\rangle$. If such a tuple does not exist, it runs the above algorithms to generate keys for user ID. If such a tuple exists and $ID \neq Bob$ then it is easy to observe that S is able to compute the corresponding session key (because it must know the corresponding secret key). Otherwise, if such a tuple exists and the queried identity is Bob, then S can compute the session key as well, but this case is slightly more complicated.

  This is where the simulator needs the oracle $\mathsf{DH}(y, \cdot, \cdot)$. Indeed, in this case S does not know the entire secret key. Recall that the correct session key is $H_2(z_1, z_2)$ with $z_1 = P^{s^*}$ and $z_2 = P^{t^*}$ where (s^*, t^*) is the secret key of Bob. Observe that z_1 is the Diffie-Hellman of $g^{s^*} = r^* y^{H_1(B, r^*)}$ and P. The simulator knows t^* and k^* (the discrete log of r^* in base g). S can compute $z_2 = P^{t^*}$ and then check if $\overline{H_2}$ contains a tuple $\langle z_1, z_2, R\rangle$ such that $\mathsf{DH}(y, P, \overline{z_1}) = $ "yes'' where $\overline{z_1} = (z_1/P^{k^*})^{H_1(B, r^*)^{-1}}$. If S finds a match then it outputs the corresponding R as the queried session key. Otherwise it generates a random $\rho \xleftarrow{\$} \{0, 1\}^\ell$ and gives it as response to the adversary. Later, for each query (z_1, z_2) to H_2, if (z_1, z_2) satisfies the equation above it answers with ρ. This makes oracle's answers consistent.

Challenge At some point the adversary sends a challenge identity ID^*. If ID^* is different from Bob the simulator outputs "Abort" and terminates the simulation. Otherwise it searches $\mathsf{PubKeyList}$ for $\langle ID^*, r^*, u^*\rangle$. If such a tuple does not exist, it runs the above algorithm to obtain it. Then S sets the pseudonym $P^* = T$, selects a random string $R^* \xleftarrow{\$} \{0, 1\}^\ell$ and gives (P^*, R^*) to the adversary. Observe that the correct session key is $R^* = H_2(z_1, z_2)$ where $z_1 = (P^*)^{s^*}$, $z_2 = (P^*)^{t^*}$ and s^* is the partial secret key of Bob (which is not known by the simulator). It is easy to see that (z_1, r^*) is a valid forgery for the XCR signature scheme.

Phase 2 In this phase the simulator acts as in phase 1.

Guess At the end of the game $\mathcal{A}$ outputs a bit b. If the adversary has success into distinguishing a correct session key from a random one, then it must query the random oracle H_2 at the correct input (z_1, z_2). Thus S can efficiently find the pair (z_1, z_2) in the table $\overline{H_2}$ using the DH oracle and finally output (r^*, z_1) as a forgery for message ID^*.

In conclusion, if $\mathcal{A}$ breaks Type I security of our protocol with probability ϵ, then our simulator breaks the unforgeability of the XCR signature scheme with probability at least ϵ/n. $\square$

3.2.3 *Type II security*

LEMMA 2. *Our certificateless anonymous KA protocol is Type II-secure if the Strong-DH Assumption holds and H_1 and H_2 are modeled as random oracles.*

3.3 Avoiding the Strong-DH

The certificateless anonymous KA protocol given in section 3.1 is proven secure under the Strong-DH Assumption. In this section we show how to modify that protocol in such a way that its security can be based directly on CDH at the cost of one more user's public key element (and one more exponentiation).

Cash *et al.* introduced in [9] a new assumption called *Twin Diffie-Hellman* (2DH). Informally 2DH states that an adversary which is given in input random $A_1, A_2, B \in \mathbb{G}$, should not be able to compute a pair (C_1, C_2) such that C_1 and C_2 are the DH of A_1, B and A_2, B respectively. It is easy to see that this assumption is equivalent to the well known CDH. The valuable contribution of their work was to show that its "strong" version is equivalent to CDH too.

Informally the Strong-2DH assumption says that 2DH holds even in the presence of an oracle $\mathsf{2DH}(A_1, A_2, \cdot, \cdot, \cdot)$ that solves its decisional version for fixed A_1, A_2.

Therefore our idea is to modify the protocol given in section 3.1 in such a way it can be proven secure under the Strong-2DH Assumption. Then, since Cash *et al.* proved in [9] that Strong-2DH and CDH are equivalent, we obtain a protocol secure under CDH.

Since the setup and partial secret key extraction algorithms are as before, here we discuss only the remaining ones.

User's key generation A user ID that obtained a partial secret key $d_{ID} = (r, s)$ from the KGC, generates its own pair of keys as follows. It picks random $t_1, t_2 \xleftarrow{\$} \mathbb{Z}_q$ and sets $u_1 = g^{t_1}$ and $u_2 = g^{t_2}$. The public key is $\mathsf{pk}_{ID} = (r, u_1, u_2)$ while the secret key is $\mathsf{sk}_{ID} = (s, t_1, t_2)$.

A protocol session Later, when a user U wants to establish a session key with Bob holding $\mathsf{pk}_B = (r, u_1, u_2)$, it proceeds as follows. First, U selects a random $w \xleftarrow{\$} \mathbb{Z}_q$ and defines $P_U = g^w$ as its pseudonym. Then it computes the session key as $K = H_2(z, z_1, z_2)$ where $z = (ry^{H_1(B,r)})^w$, $z_1 = u_1^w$ and $z_2 = u_2^w$. Given P_U, Bob can recover the same session key by computing $z = P_U^s$, $z_1 = P_U^{t_1}$ and $z_2 = P_U^{t_2}$.

THEOREM 3. *The protocol described above is a secure certificateless anonymous key-agreement protocol under the CDH Assumption if H_1 and H_2 are modeled as random oracles.*

4. CERTIFICATELESS ONION ROUTING

In this section we present our onion routing protocol. It follows the same paradigm of pairing-based onion routing in the identity-based setting from [23] with the difference that our protocol is defined in the certificateless encryption setting. As further discussed in section 4.2, our scheme is computationally more efficient than pairing-based onion routing, for several reasons: mostly that expensive pairings computations are avoided, and forward secrecy is obtained with rekeying operations that do not involve the KGC

We present our certificateless onion routing protocol by describing the following phases:

Setup In this phase the service provider acts as a KGC for our anonymous KA protocol. It generates the public parameters and issues partial secret keys for the onion routers.

Key Generation Once an onion router OR_i has obtained a partial secret key d_i, it generates a pair of keys $(\mathsf{pk}_i, \mathsf{sk}_i)$ from d_i and publishes pk_i in the public list of the system.

To achieve forward secrecy this pair of keys has a limited validity period and thus each onion router will repeat this phase after such a period is expired. However we notice that this process does not involve the KGC but can be repeated by OR_i using the same d_i. Further discussions about forward secrecy are postponed to the following section.

Circuit construction When a user wants to build a circuit, he chooses an ordered sequence of n onion routers $OR_1, \ldots, OR_n$ at random among those present in the public list. He gets their public keys and then runs the (non-interactive) anonymous KA protocol described in section 3.1 to establish a session key with each of them. Finally he creates an onion

$$P_1, \{OR_2, P_2, \{\ldots\{OR_n, P_n, \{\emptyset\}_{K_n}\}\ldots\}_{K_2}\}_{K_1}$$

where $\{m\}_{K_i}$ means that the message m is symmetrically encrypted using the session key K_i. As one can see, an onion is defined by a pair (P, C) where P is a pseudonym and C is a ciphertext.

The user sends the onion to the first onion router in the circuit. When OR_i receives an onion (P_i, C_i) it proceeds as follows. First it recovers the session key K_i using its secret key and the pseudonym P_i and then it uses K_i to decrypt C_i. It gets back a triple $(OR_{i+1}, P_{i+1}, C_{i+1})$ and sends (P_{i+1}, C_{i+1}) to the next onion router OR_{i+1}.

When an onion router gets $\emptyset$ from decrypting a ciphertext it means that it is the last router of the circuit. Therefore it sends back a confirmation message $\{Ack\}_{K_n}$ to the previous router in the circuit. When an onion router OR_i receives a confirmation message, it encrypts it using K_i and sends it to the previous router. In order to do this each router has to store the session keys and the pseudonyms. This is useful also to prevent replay attacks.

Finally, when the user receives a confirmation message, he verifies its validity by decrypting it using the session keys $K_1, \ldots, K_n$.

If the circuit construction was successful, the user can later use the circuit for his communication as in the other onion routing protocols. In particular the user sends onions through the circuit encrypting layer i with the key K_i.

4.1 Security of certificateless onion routing

In this section we show that the certificateless onion routing protocol described in the previous section is secure according to the definition given in section 2.1.1. To do this we show separately that it satisfies the following properties.

Cryptographic Unlinkability Since this property is implied by the IND-CPA security of the symmetric encryption scheme (see section 2.1.1), our protocol automatically satisfies cryptographic unlinkability if the employed symmetric encryption scheme is IND-CPA secure.

Integrity and Correctness Our protocol trivially achieves integrity and correctness. If n is the upper bound on the number of routers in the circuit, then an onion containing more than n layers of encryption can be easily recognized by the first router looking at its length. Correctness is satisfied for the construction and the correctness of the anonymous key agreement protocol.

Onion Security It is easy to observe that in our protocol onion security directly follows from the security of the certificateless anonymous key agreement protocol. In our protocol the adversary is allowed to corrupt parties and learn their secret keys, modify and/or inject public keys of users (e.g. it can replace data in the public list of the system) and ask for decryption of onions (e.g. it can submit an onion to a node in the circuit and then capture its output). It is easy to see that these capabilities can all be managed using those of an adversary in the Type I and Type II security games (see section 3).

In addition Type II security implies that our onion routing protocol is resistant even against the corruption of the KGC. In this case an attacker that recovers the KGC's master secret key should not be able to learn the session keys computed before the corruption has occured. As further discussed in section 4.1.1 this is what allows to avoid changing the KGC master keys.

Circuit Position Secrecy The protocol described in section 4 is not resistant to the generic attack (showed by Camenisch and Lysyanskaya in [5]) that allows to learn the position in the circuit of a ciphertext's recipient. Indeed one can look at the length of a ciphertext to derive such information.

The "basic" pairing-based onion routing protocol given in [23] has the same drawback. Still we notice that we can apply the same technique proposed by Kate *et al.* to make our protocol resistant to this attack at the cost of much computation and longer ciphertexts.

4.1.1 Forward secrecy

Forward secrecy informally guarantees that after a secure communication is established by two parties and ephemeral values are erased by these parties, it should be infeasible for an attacker to break the security of the past communication even if it corrupts both the parties and learns their secret keys. In this case "security" refers to the usual indistinguishability notion for session keys.

Kate *et al.* [23] showed that a single-pass protocol cannot achieve this notion since an adversary that corrupts an onion router and learns its secret key, is always able to compute a session key for any pseudonym. A solution for this issue is changing onion routers' keys. In this case we say that a protocol achieves *immediate* forward secrecy if it is secure against an adversary that corrupts an onion router *immediatly* before its keys are changeds, namely within a key validity period τ. Otherwise, if forward secrecy holds only after onion routers' keys are changed, we say that the protocol satisfies *eventual* forward secrecy.

Our protocol achieves *eventual* forward secrecy by setting a short key validity period (e.g. an hour) that prevents such attacks in practice.

The pairing-based onion routing protocol of Kate *et al.* [23] suffers from the same problem, but their solution is more complicated and less efficient than ours. Indeed Kate *et al.* suggest to have two distinct validity periods: one for the KGC's keys and one for the onion routers' keys, where the former is longer than the latter. We notice that the KGC is involved even when onion routers' keys are changed because it has to generate them. This implies a significant computational effort for the KGC.

In contrast our solution, as one can see in section 4, has only one validity period for onion routers' keys. Such keys are generated by onion routers on their own and thus we do not require the KGC to perform any additional computation. In addition our protocol provides security even against a (passive) attacker that corrupts the KGC.

4.2 Efficiency and comparisons with other protocols

To complete the analysis of our certificateless onion routing protocol, we discuss in this section its efficiency when implemented with specific parameters and compare these results with those obtained by the pairing-based onion routing protocol (PB-OR) of Kate *et al.* [23] and Tor (implemented using its specification [14]).

In our comparisons we analyze the cost of building a circuit of length n, from the perspective of both a user and an onion router. We consider security parameters of 80 and 128 bits and stress that the latter should be considered the standard for an adequate level of security (see [35, 36] for more informations about recommended key sizes).

Before presenting our results we briefly describe which operations are involved during the process of building a circuit in each of the three protocols.

In Tor the circuit is constructed using the telescoping technique in which a node establishes secure channels with the onion routers using a Diffie-Hellman (DH) key exchange [13]. More precisely the user sends to each onion router the DH parameter encrypted with RSA. Therefore a user performs 1 RSA encryption and 2 exponentiations for each of the n onion routers while each onion router performs 1 RSA decryption and 2 exponentiations. Using 80 bits of security RSA is instantiated with a 1024-bits modulus and Diffie-Hellman uses a 1024-bit finite field. In particular (according to the specifications given in [14]) RSA uses a fixed exponent 65536 while DH is optimized with exponents of 360 bits and generator 2. When we enhance the security level to 128 bits, the RSA problem uses a 3072 bits modulus and the same holds for the size of the DH finite-field.

To measure the cost of the pairing-based onion routing protocol (PB-OR) we assume to implement it in elliptic curves groups that provide the best efficiency for their case. At both the security levels we used a Type A curve (in the PBC library [25]) providing groups in which a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is defined. Kate *et al.* suggested in [23] that a user can optimize this phase by pre-computing a pairing for each onion router (as a function of the public parameters and the onion router's identity) but this process must be repeated every time the KGC's keys change (e.g. every day). Then the user must compute n exponentiations in the group $\mathbb{G}$ and n exponentiations in $\mathbb{G}_T$. On the other hand, each onion router must compute only one pairing. Although such curves provide good computational efficiency (especially for pairing computation), the same does not hold from the point of view of the space required to represent group elements. Indeed each element of $\mathbb{G}$ needs 512 bits at an 80-bits security level and 1536 bits when 128 bits of security are chosen.

In our protocol we can use the same optimization of [23] and pre-compute the value $ry^{H_1(ID,r)}$ for each router ID (where r is in its public key). Moreover in our case this pre-computation does not need to be repeated since KGC's keys do not change. A user must compute 3 exponentiations for each of the n onion routers. One of these exponentiations, $z_2 = u^w$, cannot use pre-computation since u changes frequently. On the other side an onion router performs 2 exponentiations to compute the session key. For efficiency reasons we implemented our protocol using elliptic curves groups. In this case the operations were implemented with the PBC library as well, but here we used Type F curves which

particularly fits our case because they provide short representation for $\mathbb{G}_1$ (160 or 256 bits with 80 or 128 bits of security respectively) and also efficient exponentiations in this group[3].

In our efficiency comparisons we also consider our certificateless onion routing protocol when instantiated with the anonymous KA protocol of section 3.3. From a computational perspective this protocol requires one more exponentiation to compute the session key, but it has the advantage of being based on the standard CDH assumption.

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. A summary of these costs is in Table 1.

Operation	Time (ms)	
	80-bits	128-bits
RSA Enc	0.1	0.7
RSA Dec †	3.3	67.5
Exp (Tor)	1.8	12.9
Exp. in $\mathbb{G}$ †	0.9	7.5
Exp. in $\mathbb{G}_T$ †	0.2	1.8
Exp. in $\mathbb{G}_1$	1.7	4.1
Exp. in $\mathbb{G}_1$ †	0.2	0.5

† These costs are obtained using precomputation.

Table 1: Summary of costs of operations (in ms).

Table 2 contains the total costs of building a circuit of length n respectively in Tor, PB-OR and the two instantiations of our protocol: the one using the anonymous KA of

[3]This kind of curves also admit a pairing operator $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, which is not particularly efficient to compute, but this is not of interest in our protocol as we use only exponentiations in $\mathbb{G}_1$.

section 3.1 based on Strong-DH (we call it CL-OR) and the other one using the anonymous KA of section 3.3 based on CDH (we call it 2-CL-OR). Moreover we consider two levels of security: 80 and 128 bits.

Protocol		Total cost (ms)	
		80-bits	128-bits
Tor	User	$3.7n$	$26.2n$
	OR	6.9	93.3
PB-OR	User	$1.1n$	$9.3n$
	OR	3.9	57.3
CL-OR	User	$2.1n$	$5.1n$
	OR	3.4	8.2
2-CL-OR	User	$3.8n$	$9.2n$
	OR	5.1	12.3

Table 2: Total costs to build a circuit of length n.

If we consider a security level of 80 bits our protocol CL-OR is more efficient than Tor and PB-OR when it is run both in the user and the onion router. At this level of security 2-CL-OR is slightly slower than CL-OR, but it is based on a more standard assumption.

The situation changes totally in favor of our certificateless onion routing protocols when 128 bits of security are considered. In particular we significantly improved the cost of running the protocol in an onion router as we need (in CL-OR) 8.2 ms against 57.3 ms of PB-OR and 93.3 ms of Tor. We also stress that efficiency in onion routers is more important as they are typically required to perform much more operations than the user. Moreover, since Tor is in the PKI setting, it should also include certificate verifications that are not considered in our comparisons.

Another interesting aspect that should be addressed in a comparison is the bandwidth. First of all, since we follow the same paradigm of Kate *et al.* [23], we achieve the same gain in the number of exchanged (and simmetrically-encrypted) messages. While in Tor this number is quadratic in n, in PB-OR and in ours it is only linear. Second, in our protocols we need fewer bits to represent group elements (i.e. pseudonyms): 160 and 256 bits against 512 and 1536 bits of PB-OR respectively at an 80 and 128 bits security level.

Other interesting aspects of our protocol regard forward secrecy and the workload of the KGC. The main point is that we do not require the KGC to change keys. This has several advantages. On the users' side it means that they have to obtain KGC's parameters only once and the same holds for onion routers requesting partial secret keys. On the other hand, from the KGC's perspective, we obtain a significantly smaller computational load since it does not need to repeat the users key generation phase many times (as in [23]). In addition our protocol remains secure against a passive adversary that corrupts the KGC.

Acknowledgements

The authors would like to thank Nikita Borisov for suggesting the problem and pointing them to [22].

5. REFERENCES

[1] M. Abdalla, M. Bellare and P. Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. *In proceedings of CT-RSA 2001*, LNCS vol. 2020, pp. 143-158.

[2] S. Al-Riyami and K. Paterson. Certificateless public key cryptography. *Advances in Cryptology – ASIACRYPT 2003*, 2003, LNCS vol. 2894, pp. 452-473.

[3] Dan Boneh, Xavier Boyen. Short Signatures without Random Oracles. *Advances in Cryptology – Eurocrypt 2004*, LNCS vol. 3027.

[4] Dan Boneh, Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM J. Comput.* 32(3): 586-615 (2003) (Also in CRYPTO 2001.)

[5] J. Camenisch and A. Lysyanskaya. A Formal Treatment of Onion Routing. *Advances in Cryptology – CRYPTO 2005*, LNCS vol. 3621, pp. 169-187.

[6] R. Canetti. Universally Composable Security: A new paradigm for cryptographic protocols. *In proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, 2001, pp. 136-145.

[7] R. Canetti, H. Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. *Advances in cryptology – EUROCRYPT 2001*, LNCS vol. 2045, pp. 453-474

[8] R. Canetti, H. Krawczyk. Universally Composable Notions of Key Exchange and Secure Channels. *Advances in cryptology – EUROCRYPT 2002*, LNCS vol. 2332, pp. 337-351

[9] D. Cash, E. Kiltz and V. Shoup. The Twin Diffie-Hellman Problem and Applications *Advances in cryptology – EUROCRYPT 2008*, LNCS vol. 4965, pp. 127-145.

[10] D. Chaum. Untraceable Electronic Mail, return address and digital pseudonyms. *Communications of the ACM*, 24(2), pp. 84-88, 1981.

[11] W. Dai. PipeNet 1.1 http://www.weidai.com/pipenet.txt

[12] A. Dent. A Survey of Certificateless Encryption Schemes and Security Models. *in International Journal of Information Security*, 2008, vol. 7, n. 5, pp. 347-377.

[13] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 1976, vol. 22, n. 6 , pp. 644-654

[14] R. Dingledin and N. Mathewson. Tor Protocol Spefication. 2008 http://www.torproject.org/svn/trunk/doc/spec/tor-spec.txt

[15] R. Dingledin, N. Mathewson and P. Syverson. Tor: The Second-Generation Onion Router. *In proceedings of the 13th USENIX Security Symposium*, 2004, pp. 303-320.

[16] A. Fiat and A. Shamir How to Prove Yourself: Practical Solutions of Identification and Signature Problems. *Advances in cryptology – CRYPTO 1986*, LNCS vol. 263, pp. 186-194

[17] D. Fiore and R. Gennaro. Making the Diffie-Hellman Protocol Identity-Based. Cryptology Eprint Archive, Report 2009/174. http://eprint.iacr.org/2009/174.

[18] M. Freedman and R. Morris. Tarzan: A Peer-to-Peer Anonymizing Networ Layer. *In proceedings of the 9th ACM Conference on Computer and Communication Security (CCS 2002)*, pp. 193-206.

[19] I. Goldberg. On the Security of the Tor

Authentication Protocol. *In proceedings of the 6th Workshop on Privacy Enhancing Technologies (PET 2006)*, 2006, LNCS vol. 4258, pp. 316-331.

[20] D. Goldschlag, M. Reed and P. Syverson. Hiding Routing Informations. *In proceedings of the First International Workshop on Information Hiding*, 1996, LNCS vol. 1174, pp. 137-150.

[21] D. Goldschlag, M. Reed and P. Syverson. Onion Routing for Anonymous and Private Internet Connections. *Communications of the ACM*, 1999, 42(2), pp. 84-88.

[22] A. Kate, G. Zaverucha and I. Goldberg. Pairing-Based Onion Routing. *In the proceedings of the 7th Privacy Enhancing Technologies Symposium (PETS 2007)*, 2007, LNCS vol. 4776, pp. 95-112.

[23] A. Kate, G. Zaverucha and I. Goldberg. Pairing-Based Onion Routing with Improved Forward Secrecy. *In ACM Transactions on Information and System Security*, 2009.

[24] Hugo Krawczyk. HMQV: A High-Performance Secure Diffie-Hellman Protocol. *Advances in cryptology – CRYPTO 2005*, LNCS vol. 3621, pp. 546-566

[25] B. Lynn. PBC: The Pairing-Based Crypto Library. http://crypto.stanford.edu/pbc

[26] B. Möller. Provably Secure Public Key Encryption for length-preserving chaumian mixes. *In proceedings of CT-RSA 2003*, LNCS vol. 2612, pp. 244-262.

[27] L. Øverlier and P. Syverson. Improving Efficiency and Simplicity of Tor Circuit Establishment and Hidden Services. *In the proceedings of the 7th Privacy Enhancing Technologies Symposium (PETS 2007)*, 2007, LNCS vol. 4776, pp. 134-152.

[28] D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 13(3):361-396 (2000).

[29] M. Reed, P. Syverson and D. Goldschlag. Anonymous Connections and Onion Routing. *IEEE Journal on Selected Ares in Communications*, 16, 4, pp. 482-494.

[30] M. Renhard and B. Plattner. Introducing MorphMix: Peer-toPeer based Anonymous Internet Usage with Collusion Detection. *In The Workshop on Privacy in the Electronic Society (WPES 2002)*, ACM, pp. 91-102.

[31] Adi Shamir Identity-Based Cryptosystems and Signature Schemes *Advances in Cryptology – Proceedings of CRYPTO '84*, 1985, pp. 47-53

[32] V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *Advances in Cryptology – Proceedings of EUROCRYPT '98*, 1998, LNCS 1403.

[33] R. Sakai, K. Ohgishi and M. Kasahara. Cryptosystems based on pairing. *In Symposium on Cryptography and Information Security*, Okinawa, Japan, 2000.

[34] C.P. Schnorr. Efficient identification and signatures for smart cards. *Advances in Cryptology – CRYPTO '89*, 1989, LNCS vol. 435, pp. 239-252

[35] 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.

[36] 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

APPENDIX

A. COMPUTATIONAL ASSUMPTIONS

In the following assume $\mathbb{G}$ to be a cyclic multiplicative group of order q where q is a ℓ-bit long prime. We assume that there are efficient algorithms to performe multiplication and membership test in $\mathbb{G}$. Finally we denote with g a generator of $\mathbb{G}$.

ASSUMPTION 1 (COMPUTATIONAL DIFFIE-HELLMAN [13]). *We say that the Computational Diffie-Hellman (CDH) Assumption (for $\mathbb{G}$ and g) holds if for any probabilistic polynomial time adversary $\mathcal{A}$ the probability that $\mathcal{A}$ on input $(q, \mathbb{G}, g, g^a, g^b)$ outputs C such that $C = g^{ab}$ is negligible in ℓ. The probability of success of $\mathcal{A}$ is taken over the uniform random choices of $a, b \in \mathbb{Z}_q$ and the coin tosses of $\mathcal{A}$.*

The CDH Assumption has a *Decisional* version in which no adversary can actually *recognize* the value g^{ab} when given g^a, g^b. In our proofs we are going to need the ability to perform such decisions when one of the two elements is fixed, while still assuming that the CDH holds. The assumption below basically says that the CDH Assumption still holds in the presence of an oracle $\mathsf{DH}(A, \cdot, \cdot)$ that solves the decisional problem for a fixed A[4].

ASSUMPTION 2 (STRONG-DH ASSUMPTION [1]). *We say that the Strong-DH Assumption holds (for $\mathbb{G}$ and g) if the CDH Assumption holds even in the presence of an oracle $\mathsf{DH}(A, \cdot, \cdot)$ that on input two elements $\hat{B}, \hat{C}$ in the group generated by g, outputs "yes" if and only if $\hat{C}$ is the Diffie-Hellman of A and $\hat{B}$.*

[4]We remark that in recent papers the name strong Diffie-Hellman assumption was used to denote a different conjecture defined over bilinear groups [3]. In this paper, we refer to the original terminology from [1]