Des. Codes Cryptogr. (2013) 67:245–269
DOI 10.1007/s10623-011-9604-z

# Anonymity and one-way authentication in key exchange protocols

**Ian Goldberg · Douglas Stebila · Berkant Ustaoglu**

Received: 6 May 2011 / Revised: 11 September 2011 / Accepted: 19 December 2011 /
Published online: 14 January 2012


**Abstract** Key establishment is a crucial cryptographic primitive for building secure communication channels between two parties in a network. It has been studied extensively in theory and widely deployed in practice. In the research literature a typical protocol in the public-key setting aims for *key secrecy* and *mutual authentication*. However, there are many important practical scenarios where mutual authentication is undesirable, such as in anonymity networks like Tor, or is difficult to achieve due to insufficient public-key infrastructure at the user level, as is the case on the Internet today. In this work we are concerned with the scenario where two parties establish a private shared session key, but only one party authenticates to the other; in fact, the unauthenticated party may wish to have strong anonymity guarantees. We present a desirable set of security, authentication, and anonymity goals for this setting and develop a model which captures these properties. Our approach allows for clients to choose among different levels of authentication. We also describe an attack on a previous protocol of Øverlier and Syverson, and present a new, efficient key exchange protocol that provides one-way authentication and anonymity.

**Keywords** Key exchange · One-way authentication · Anonymity · Tor network · Protocols · Security models

**Mathematics Subject Classification (2000)** 94A60 Cryptography

Communicated by C. Cid.

I. Goldberg
Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada
e-mail: iang@cs.uwaterloo.ca

D. Stebila (✉)
Information Security Institute, Queensland University of Technology, Brisbane, QLD, Australia
e-mail: stebila@qut.edu.au

B. Ustaoglu
Faculty of Engineering and Natural Sciences, Sabancı University, Istanbul, Turkey
e-mail: bustaoglu@cryptolounge.net

## 1 Introduction

*Authentication* is one of the core security properties that many cryptographic protocols aim to provide: one party wishes to be convinced of the identity of another party. But in many contexts, the opposite property, *anonymity* is equally important: one party wishes for no one to be able to determine her identity.[1] Anonymity is an enabling technology for privacy, whether for normal citizens going about their day-to-day lives or for dissidents and whistleblowers putting themselves in danger because of the information they transmit.

In practice, the seemingly opposite properties of anonymity and authentication go hand-in-hand: Alice, wishing to remain anonymous, wants to communicate with Bob and be assured of Bob's identity. We see this in a number of examples on the Internet today:

– In the Tor anonymity network [15,37], an anonymous client node establishes an encrypted Tor circuit through authenticated intermediate relay nodes.
– In secure web (HTTPS) transactions, the most common mode of operation has an unauthenticated client communicating with an authenticated server.

There can be various reasons for clients to be unauthenticated. It may be that the client is not trying to gain access to any resources that require authentication but is indifferent to its anonymity, or it may in fact be that the client wishes very strongly to be anonymous.

Authenticated key exchange (AKE) is one of the most important cryptographic constructs and is used to establish an authenticated and confidential communication channel. Existing approaches to two-party key exchange have emphasized mutual authentication, in which both parties authenticate themselves to their peer. The use of one-way authenticated protocols raises an important question: what is the motivation for an authenticated server to use an encrypted channel to communicate with an unauthenticated client? We will argue that in some sense, the security provided with respect to confidentiality is upper-bounded by the security provided with respect to authentication. From the server's perspective there may be no motivation to encrypt the channel with an unauthenticated client, but servers may still wish to go to the computational effort of providing such encrypted connection out of sympathy for the client's security goals or for other external reasons.

Relaxing the authentication requirements in key exchange from mutual to one-way authentication also permits additional security properties, such as anonymity. Some previous approaches to key exchange have permitted *identity hiding*, in which the identity of a party is never communicated in the clear, but still eventually becomes known to the peer. *Anonymity*, on the other hand, means that a party's identity is kept secret even from its peer, and is an important privacy tool. A few protocols have been proposed that offer some combination of one-way authenticated key exchange and anonymity, but a more thorough, formal and unified approach is merited.

The cryptographic literature focuses almost entirely on mutually authenticated key exchange, usually in the two-party setting. Security models such as those by Canetti and Krawczyk [6] and LaMacchia et al. [22] assume all parties have certificates. Thus session peers are indistinguishable in the sense that each peer can contribute static key pairs and certificates to the key establishment. However, this assumption does not reflect real-world practice. Of the millions of key agreement sessions that happen each day on the Internet, only a tiny proportion involve mutual public key authentication; the vast majority, especially

[1] An important related property is *unlinkability*, meaning that no one can link one party's various conversations to each other. See [32] for a taxonomy of anonymity-related definitions. We will see that, in the context of key exchange, unlinkability and anonymity are in a sense equivalent.

using TLS on the web, involve only one certificate authenticating the server's identity to the client, while the client provides no certificate authentication (at least in the cryptographic key agreement protocol). This is not a surprising fact. Certificates that bind static public keys to identities require a public key infrastructure (PKI), which can be costly to deploy and difficult to manage; while many big corporations can adopt certificates, average users seldom do. Moreover, users tend to move between computers—at home, at work, in Internet cafés—without carrying along private keys and certificates. Even system administrators rarely, if at all, carry private-public key pairs, but nonetheless frequently perform remote maintenance. These scenarios are often encountered in practice, but are not captured by the aforementioned models.

Password-authenticated key exchange (PAKE) [3,4] does treat clients and servers differently, and passwords can be easier to use than certificates (though passwords have their own usability and security challenges). Still, PAKE aims to provide mutual authentication for peers, which does not always match a system's needs.

In a number of situations, only one party needs, or can, or wants to be authenticated to the other. Fiore et al. [16] point out that, for the identity-based setting, a mutually authenticated key exchange protocol can be transformed into a one-way protocol by assigning a "dummy" public key / private key pair that is shared among the one-way parties. A similar technique can be applied in the non-identity-based setting. However, one-way AKE need not just be a limited form of mutually authenticated key exchange. It can, for example, pave the way for features like anonymity. The generic construction of Fiore et al. provides no anonymity assertions, and indeed it is not hard to imagine a reasonable protocol or implementation in which nonces are generated by hashing previous values; with a reasonable hash function, this would still provide security, but not unlinkability and hence not anonymity.

*Contributions.* In this paper, we consider the problem of one-way authenticated key exchange and how it relates to anonymity. We provide an intuitive set of goals that such protocols should achieve, including confidential session key agreement, one-way authentication, and anonymity. We then present a formal model that captures these goals.

Our work differs from existing approaches to models for authenticated key exchange because it provides a means to achieve not only mutual but also only one-way authentication. We provide a security model that distinguishes between sessions that authenticate peers and sessions that do not: we are concerned with the secrecy of session keys that have authenticated peers. We also incorporate in our model the ability for an adversary to predict a party's behaviour by learning public keys in advance. This is a subtle issue as there are protocols—see [24] for further discussion—where leaking public information before the protocol run commences makes the difference between security and insecurity. Our security model differs from existing authenticated key exchange security models such as CK01 [6] and eCK [22] in that it addresses one-way, not mutually authenticated, sessions, and is more general than the specialized security definition of Goldberg [17] for the one-way Tor Authentication Protocol.

In addition to one-way AKE security, our model also provides anonymity definitions, including anonymity with respect to peers and unlinkability of sessions, and we show that unlinkability is equivalent to anonymity. Anonymity with respect to peers is a stronger property than the identity hiding property present in several existing AKE protocols, such as JFKi [1], where the client's identity is hidden from eavesdroppers but eventually known to the peer.

To motivate the need for formal models and proofs, we analyze a key exchange protocol proposed by Øverlier and Syverson [31] for use in the Tor anonymity network[2] and show that this protocol is insecure: it suffers from an authentication flaw allowing adversaries to easily impersonate honest servers to honest clients. As a remedy, we introduce an efficient new one-way authenticated key exchange protocol and show that it provides security and anonymity in our model. Our protocol is suitable for use in a variety of settings, including the establishment of circuits in the Tor network.

## 2 Related work

*Authenticated key agreement.* Key agreement is an important cryptographic primitive that has been extensively studied, especially in the two-party authenticated setting [2,5–7,22,24]. However, only a few protocols have considered the problem of one-way authentication. Goldberg [17] gave a specialized one-way AKE security definition for the Tor authentication protocol. Øverlier and Syverson [31] proposed some alternative protocols for circuit establishment in Tor but without any security arguments. We analyze one of their protocols in Sect. 3 and demonstrate it does not prevent server impersonation. Kate et al. [19] describe an identity-based anonymous authenticated key exchange protocol but with a limited session key secrecy definition based on key recovery, not indistinguishability. Morrissey, Smart, and Warinschi [26] analyzed the security of the Transport Layer Security (TLS) protocol in the context of one-way authentication, but with specialized security definitions.

*Identity hiding and anonymity.* Some key exchange protocols [1,8,9,20] aim to provide *identity hiding*, in which the identity of one party remains hidden during communication, though the identity becomes available to the peer by the end of the protocol. The Canetti-Krawczyk post-specified peer model [7] and Menezes-Ustaoglu model [24] encompass key exchange protocols in which the identity of the peer may not be available at the onset of the protocol.

Other key exchange protocols [10,17,19,31,33,34] aim to give *anonymity*, in which even the peer does not learn the long-term identity of the party. This is an important goal for practical applications such as the Tor anonymity network [15]. Most of the references above do not analyze the anonymity properties of their protocols in a formal manner, although a few do. Shoup [34] defines anonymity in the context of the simulation framework for key exchange security, as opposed to the indistinguishability framework of, for example, Canetti-Krawczyk [6], which has now become more commonplace for analyzing key agreement protocols. Kate et al. [19] provide the only other one-way authenticated key exchange protocol with formal anonymity goals, although their definition is specialized for the protocol in question. In this paper, we present a generic anonymity definition, suitable for analyzing a wide variety of protocols, in a framework similar to the key exchange security framework of Canetti-Krawczyk [6].

Some protocols [11] provide *deniability*, where it cannot be conclusively proven that a party participated in a key exchange session. This differs from anonymity in that a deniable protocol may still leak information about the parties involved in its normal operation.

*One-way vs One-flow.* We make special note of the difference between *one-way* AKE and *one-flow* AKE. One-flow AKE protocols are designed to establish a session key using a single message from the client to the server. It can provide mutual authentication by using two static

[2] We note that Øverlier and Syverson's protocol has not yet been *used* in the Tor anonymity network; thus, this is not an attack on the deployed Tor network.

keys (one each from the client and the server) and one ephemeral key (from the client). In contrast, one-way AKE can use one static key (from the server) and two ephemeral keys (one each from the client and the server), but provides no authentication to the server. Although one can try to view a one-way AKE protocol as the complement of a one-flow AKE protocol, switching ephemeral and static keys, the security properties are substantially different.

## 3 Cryptanalysis of a protocol of Øverlier and Syverson

Øverlier and Syverson [31] proposed a number of one-way authenticated key exchange protocols to be used for circuit establishment in the Tor anonymity network. They presented several protocols, culminating in the so-called *fourth protocol*, which aims to provide reduced computational requirements, key establishment with immediate forward secrecy, and implicit one-way authentication. The protocols did not have any accompanying security argument. We had originally set out to prove the fourth protocol secure in our model, but instead discovered an attack against the protocol.

*Fourth protocol.* Let $g$ be a generator of a group of prime order $q$. Suppose a client $\hat{A}$ wishes to establish an encrypted session with server $\hat{B}$ with long-term Diffie-Hellman private key $b$ and public key $B = g^b$. The client and server exchange random ephemeral Diffie-Hellman public keys $X = g^x$ and $Y = g^y$, respectively, and the session keys are derived from the cryptographic key material $k$, which is computed by the client as $(BY)^x$ and by the server as $X^{b+y}$.

*Attack.* Our attack allows an adversary $\mathcal{M}$ to impersonate an honest server to an honest client. The adversary is able to determine the session key and thereby convince honest clients that it is the honest server simply from having learned the server's long-term public key. This is essentially a full break of the protocol.

The attack proceeds as follows. Suppose $\mathcal{M}$ knows server $\hat{B}$'s long-term public key $B$. An honest client $\hat{A}$ sends its ephemeral public key $X$ to $\hat{B}$, which $\mathcal{M}$ intercepts. $\mathcal{M}$ computes $B^{-1}$, selects a value $r \leftarrow \mathbb{Z}_q$, and computes $Y' \leftarrow B^{-1}g^r = g^{r-b}$; $\mathcal{M}$ responds to the client with $Y'$. The client computes the key material $k \leftarrow (BY')^x = g^{(b+r-b)x} = g^{rx}$. The attacker $\mathcal{M}$ computes the same key material as $X^r = g^{rx}$, and hence can derive the same session keys as $\hat{A}$. This allows $\mathcal{M}$ to impersonate $\hat{B}$ to $\hat{A}$.

For the purposes of breaking an entire Tor circuit, $\mathcal{M}$ can use this technique to successively impersonate all nodes in a Tor circuit establishment by $\mathcal{M}$.

*Lessons learned.* This attack serves to underline the importance of formal security arguments for protocols and motivates our development of a security model for one-way AKE in Sect. 5. Before developing a security model, we first examine security goals in the one-way authenticated scenario and how they differ from the two-way authenticated scenario.

## 4 Understanding one-way authentication

In this section, we try to understand the motivation of various parties to engage in one-way AKE protocols. We then investigate the relation of one-way AKE protocols to key confirmation and public-key encryption.

### 4.1 Secrecy without authentication

If authentication is not required, is secrecy? Consider the following examples:

1. Readers act upon the perceived reputation of a news source, so reputation is a valuable commodity for utility providers such as journalists. *Provider-to-user* authentication achieves both reputation damage prevention for the provider and source integrity for users. No further authentication is required and since the information is public, channel secrecy is not required and does not affect the actions of either party.
2. A growing community achieves online privacy via the Tor network [37]. Tor hides user actions by relaying encrypted connection through authenticated intermediate servers, without requiring any user authentication. To avoid Internet surveillance users must authenticate their Tor entry point and keep the subsequent communication encrypted. However, Tor is available even to users who after utilizing Tor disclose all their actions.
3. Until recently most search engines redirected HTTPS to HTTP requests thus denying secrecy to clients.[3] On the server side HTTPS connections require extra valuable resources while the actual response content is not affected [18]. On the user end, however, encryption availability likely affects the content of sensitive queries.
4. Patients requiring medical advice may wish to do so anonymously, while still ensuring the confidentiality of their request and assurance that the medical advice received comes from an authentic, qualified source.

For a single communication channel these examples illustrate that the authenticated party provides the same services irrespective of secrecy. On the other hand, the party verifying the authentication could alter its behaviour depending on the availability of secrecy. Effectively, the party with an incentive to perform authentication on the other also has an incentive for the channel to be encrypted. In that sense, secrecy assurances stronger than authentication assurances are not well-motivated in an individual communication session. However, parties not receiving authentication can still have secrecy-related goals:

– Doctors may be required by law to preserve patient-doctor confidentiality. Even in the case of an unauthenticated patient, doctors may prefer an encrypted channel over an unencrypted channel to ensure only the session initiator can read the responses. This is subtly different than the standard secrecy definition and is perhaps better described as exclusivity. Even without this encrypted channel, however, the primary actions of the doctor—the medical advice given—will be unaltered.
– An Internet service provider (ISP) could make additional revenue by either replacing ads on returned search queries with its own ads [36], or by observing and selling user trends [35]. For search engines that accept unauthenticated connections and obtain revenue from selling advertising space, end-to-end secrecy and message integrity could protect their business model from unfair competition. Arguably, however, the search engine's goal here is not the secrecy of any separate connection, but the security of a large number of connections overall. Hence other aggregate-oriented definitions of secrecy may be relevant here.

In this work, we focus on authentication and secrecy goals for individual connections, and therefore to parties that receive authentication assurances. This excludes alternative goals such as connection exclusiveness and aggregate security, which are different targets and could potentially be achieved via alternate methods.

[3] As of May 2010, Google started accepting HTTPS search requests [18].

### 4.2 Key confirmation and one-way authentication

Key confirmation is an integral part of many authenticated key exchange protocols. The TLS protocol [12], for example, uses `Finished` messages in both directions, which are computed as a MAC, under the master secret key, of the text "client finished" or "server finished" and the fingerprint of the transcript. Key confirmation enhances the security properties of the basic schemes. NIST SP800-56A [28] states:

> Key agreement, accompanied by key confirmation [...] can be used to provide the recipient with assurance of either the provider's current or prior possession of the static private key that is associated with a particular static public key.

Moreover, in SP800-56A key confirmation is associated with the use of a static key pair: a party provides a key confirmation message only if that party contributed a static key pair in the key agreement protocol. In the previous section we gave examples where parties not authenticating their peers do not alter their behaviour based on the presence of encryption. From the point of view that key confirmation enhances session-key secrecy and authentication properties, client-to-server key confirmation is not needed.

While there may be no cryptographic reason for key confirmation from an unauthenticated client to an authenticated server, from an *engineering perspective* the server may desire assurances that the client has computed the session key and should keep the connection open. MAC tags provide a means for the server to confirm that the peer has completed the session. However, confirmation that the unauthenticated client has successfully completed the session does not require secret-key cryptographic operations; it can be achieved simply by using a hash of all public messages. Such an approach is no worse than sending a MAC tag: in preventing any half-open-connection denial-of-service attacks, confirmation via a hash of exchanged messages is as successful as confirmation via a MAC tag. The computational cost of symmetric-key operations is negligible but minimizing cryptographic operations nevertheless reduces the overall complexity of server management by imposing weaker conditions on the environment. Consequently, the protocol we propose in Sect. 6 employs only server-to-client key confirmation and no client-to-server key confirmation.

### 4.3 Public-key encryption and one-way AKE protocols

Public key encryption can be used for one-way AKE protocols, for example by having the client encrypt a session key under the server's public key. This mechanism is widely used, for example in the RSA-based cipher suites in TLS [12, §7.4.7.1] and in the KAS1 protocol in NIST SP800-56B [29, §8.2].

Key agreement using public-key encryption provides a form of one-way authentication. NIST SP800-56B [29, §8.2.4] describes the authentication properties of KAS1 as follows:

> In each scheme included in this family, only the identifier of V (the responder) is required to be bound to a public key-establishment key. U (the initiator) has assurance that no unintended party can recover $Z$ from $C$ (without the compromise of private information).
> The responder, however, has no such assurance. In particular, V has no assurance as to the accuracy of the identifier claimed by the initiator and, therefore, has no assurance as to the true source of the ciphertext $C$.

Existing two-party AKE security models allow the adversary to select any session (as long as it is not trivially compromised) as the test session. In key agreement protocols such as

KAS1, where the responder does not obtain any assurance of its peer's identity, AKE security is hard to achieve: the adversary can create a ciphertext on its own and engage with an honest party "on behalf" of another honest party. Since the adversary can compute the session key, no secrecy assurances can be deduced and thus existing AKE definitions are not useful here.

Recall that in a typical indistinguishability-based adaptive chosen ciphertext attack (IND-CCA2) security experiment, the adversary plays a game against a party that *encrypts* messages. In other words, it is the *sender* of the message that obtains security assurances and has assurances about the identity of the receiving party. In line with the discussion in Sect. 4.1, the IND-CCA2 security definition implies no (direct) assurances for the *receiver*. We therefore adapt this idea to our security definition for one-way AKE. Our adversary can only select target test sessions that have an authenticated peer, thus overcoming the aforementioned drawback of existing models where the adversary can target any session.

Our one-way AKE model can be viewed as a refinement of authentication and secrecy notions in key agreement, but it can also be seen as an extension of IND-CCA2 security to the multi-party setting. Such a unified treatment of key establishment and public-key encryption should not come as a surprise, since a typical usage of both primitives is to establish a shared secret key between two parties that can subsequently be used with symmetric encryption to transfer larger amounts of data.

## 5 Security model

In the previous section we looked at the meaning of one-way authentication and argued that existing models leave outside their scope one-way authenticated protocols. In this section, we describe the security model and security experiments we consider relevant for one-way authenticated key exchange. The section is divided into two parts: the overall model of what protocols, parties, and the adversary *are* and how they *interact*, and then a number of security experiments in that model to describe various security properties such as one-way authenticated key exchange and anonymity.

Our model is based on the eCK model [22], but with a few changes to support one-way authentication, as well as some syntactic changes. Most importantly, we do not focus on matching conversations or transcripts but on the output of a session. The output is defined by the protocol, and it is the output that indicates which sessions should produce the same session key. The grouping of values in the session output also defines which combinations of values can be revealed by the adversary without compromising security; this notion of *partnering to a value* is the link between parties, public keys, and session outputs, and is crucial for our definitions. Finally, we provide the adversary with additional power compared to CK01 or eCK, namely the ability to learn public values before they are used (RevealNext).

### 5.1 Model description

*Parties, key pairs, and certificates.* Parties implementing protocols are probabilistic interactive Turing machines. Parties can be activated via incoming messages. The responses are outgoing messages as defined by the protocol or confirmation that certain routines were completed with either success or failure.

Each party will have in its memory many *key pairs* of the form $(x, X)$, where $x$ is a private value and $X$ is a public value. The key pairs may be generated by some algorithm specified by the protocol. A party may have two types of key pairs: *ephemeral* key pairs, which are associated with a particular session $\Psi$, or *static* key pairs, which can be used across multiple

sessions. The party may also have key pairs that have been generated but not yet used. If necessary, different types of key pairs may be permitted, for example, if a protocol uses has one type of key pair for digital signatures and another type of key pair for public-key encryption.

Each party will also have public keys $X$ that have been bound together with a party identifier $\hat{X}$ to form a *certificate* $cert_X = (\hat{X}, X)$ that it will use to authenticate other parties. A party is said to be an *owner* of a certificate $(\hat{X}, X)$ if it knows the secret key $x$ corresponding to the public key $X$.

*Protocol and sessions.* A protocol is a collection of interactive routines, along with a set of public parameters $pubparams$. During execution a protocol can request access to the party's memory. Each execution of a protocol is called a *session*, and each session has an associated *session identifier* $\Psi$ assigned by the party, which must be unique within the party. Each session has an associated *session state* where intermediate values are stored, which is updated according to the protocol specification and the incoming messages delivered to the session. Where necessary, we identify the session state for party $P$ in session $\Psi$ by $M^P_{\mathbf{state}}[\Psi]$. If a session $\Psi$ is executed within a party then we call that party the *owner* of $\Psi$.

*Session execution.* Within a session execution for the session $\Psi$, the computation may access the party's static key pairs as well as certificates of other parties. Furthermore, the session $\Psi$ can access key pairs $(x, X)$ that are not yet bound to other sessions, or direct novel pairs to be generated if no unused pairs exist; if such a pair $(x, X)$ is accessed or generated, then it becomes *bound* to $\Psi$ and becomes part of $\Psi$'s session state. After the session completes its execution, pairs $(x, X)$ that were part of the session state are deleted. During the session execution the session may also produce outgoing messages.

Once a session $\Psi$ owned by $P$ completes, it outputs a value $M^P_{\mathbf{out}}[\Psi]$ which is either $\perp$ or $(sk, pid, \mathbf{v})$, where $sk$ is a session key in a keyspace $\mathcal{K}$, $pid$ is a party identifier or the anonymous symbol $\circledast$, and $\mathbf{v} = (\mathbf{v}_0, \mathbf{v}_1, \dots)$ where each vector $\mathbf{v}_i$ is a vector of public values. (For example, $\mathbf{v}_1$ may consist of the public values contributed by party $P_1$.) Including $\mathbf{v}$ as part of the session output binds the session with the various keys used by the parties; this serves the same purpose as including a partial transcript in the session identifier in the eCK model. We use the notation $M^P_{\mathbf{out}}[\Psi].sk$ to refer to the session key for the session, and similarly for the remaining output values.

*Adversary.* The adversary is a probabilistic Turing machine taking as input $pubparams$ with access to oracles for parties $P_1, \dots, P_n$. The adversary controls all communications between the parties using the specified queries; in particular, it can delete, modify, insert, delay, and reorder messages. The adversary can direct a party to perform certain actions by sending one of the following queries to the party $P$:

– $\mathsf{Send}(params, pid) \rightarrow (\Psi, msg)$: With this query, the adversary directs the party to initiate a new key exchange session. The party initiates a new session and assigns a new session identifier $\Psi$. The response to this query includes any protocol-specific outgoing message $msg$ as well as the session identifier $\Psi$. The format of the input value $params$ is specified by the protocol and could include: (i) the protocol to be executed; (ii) the certificate(s) which the party should use to authenticate itself; (iii) the certificate(s) to use for peers in the session. The value $pid$ is the identifier of the party with whom to establish the session; if the session is meant to be established with an unauthenticated, anonymous peer, then $pid$ is the special symbol $\circledast$.

- $\mathsf{Send}(\Psi, msg) \rightarrow msg'$: This query models the standard delivery of messages. The party activates session $\Psi$ with $msg$ and returns any outgoing message $msg'$.
- $\mathsf{RevealNext} \rightarrow X$: This query allows the adversary to learn future public values. The party generates a new key pair $(x, X)$, records it as unused, and returns the public value $X$.
- $\mathsf{Partner}(X) \rightarrow x$: This query allows the adversary to compromise secret values used in the protocol computation. If the party has a key pair $(x, X)$ in its memory, it returns the private value $x$.
- $\mathsf{SessionKeyReveal}(\Psi) \rightarrow sk$: With this query, the adversary can learn session keys. This returns the secret key $M^P_{\mathbf{out}}[\Psi].sk$ for session $\Psi$.

Additionally, the adversary can generate its own public keys and certificates with the following non-party-specific query:

- $\mathsf{EstablishCertificate}(\hat{X}, X)$: The adversary registers with all parties a certificate containing public key $X$ for an unused identifier $\hat{X}$; the adversary is the owner of such a certificate.[4]

Certification can also be made an adaptive process. That is, the adversary can be allowed queries that direct a party to generate a new certificate for a pair $(x, X)$ or accept new or different peer certificates. For simplicity we do not address this issue here.

Where necessary to avoid ambiguity, we use a superscript to indicate the party to whom the query is directed, for example $\mathsf{Send}^{P_i}(\Psi, msg)$.

*Partnering.* The adversary is said to be a *partner* to a value $X$ unless $X$ was the output of a Send or RevealNext query to some party $P_i$ and $\mathsf{Partner}(X)$ was never issued to party $P_i$. Note that this implies that the adversary is partner to almost all values. For consistency, if a party generated the pair $(x, X)$ — either because it was directed by the adversary or because of a session execution — then the party is said to be a partner to $X$.

*Correct protocols.* A two-party key exchange protocol $\pi$ is said to be *correct* if, under the following conditions:

- the adversary is benign, meaning it faithfully relays all messages in a protocol run between the two parties, and
- if a party is activated with a Send query in which the peer identifier $pid$ is not $\circledast$, then that party holds a certificate corresponding to $pid$;[5]

the following hold for each run of the protocol:

- both parties accept and output the same session key $sk$ and the same vectors $\mathbf{v}$, and
- the value $pid$ in each party's output matches the partner identifiers $pid$ in the Send query with which the party was activated.

### 5.2 One-way AKE security

The goal of the adversary in the one-way AKE security experiment is to distinguish the session key of an uncompromised session from a random key. In the one-way authenticated key exchange security experiment, the adversary has access to the following additional oracle:

[4] We often equate identifiers and parties. If an identifier $\hat{X}$ and associated public key $X$ was introduced with an EstablishCertificate query, we call $\hat{X}$ a *dishonest* party; otherwise $\hat{X}$ is said to be *honest*.

[5] If certification is treated as an adaptive process, then correctness further requires that this certificate be unchanged during the run of that session.

– $\mathsf{Test}(i, \Psi) \rightarrow \mathcal{K}$: Abort if $M_{\mathbf{out}}^{P_i}[\Psi].sk = \perp$ or $M_{\mathbf{out}}^{P_i}[\Psi].pid = \circledast$. Otherwise, choose $b \xleftarrow{\$} \{0, 1\}$. If $b = 1$, then return $M_{\mathbf{out}}^{P_i}[\Psi].sk$; otherwise, when $b = 0$, return a random element of $\mathcal{K}$. Only one call to the $\mathsf{Test}$ query is allowed.

*One-way-AKE-fresh.* A session $\Psi$ at a party $P_i$ is said to be *one-way-AKE-fresh* if both of the following conditions hold:

1. For every vector $\mathbf{v}_j$, $j \geq 1$, in $M_{\mathbf{out}}^{P_i}[\Psi]$, there is at least one element $X$ in $\mathbf{v}_j$ such that the adversary is not a partner to $X$.
2. The adversary did not issue $\mathsf{SessionKeyReveal}(\Psi')$ at party $P_j$, for any $P_j$ which is the owner of a certificate for the identifier $M_{\mathbf{out}}^{P_i}[\Psi].pid$ such that $M_{\mathbf{out}}^{P_i}[\Psi].\mathbf{v} = M_{\mathbf{out}}^{P_j}[\Psi'].\mathbf{v}$.

**Definition 1** (*One-way-AKE-secure*) Let $k$ be a security parameter and let $n \geq 1$. A protocol $\pi$ is said to be *one-way-AKE-secure* if, for all probabilistic polynomial time (in $k$) adversaries $\mathcal{M}$, the advantage that $\mathcal{M}$ distinguishes a session of a one-way-AKE-fresh session from a randomly chosen session key is negligible (in $k$).

*Remark.* We emphasize that the test session can be directed only at sessions that outputs an identifier $pid$ of an authenticated party. If a session does not output any authenticated peer then the session key security of such sessions is not guaranteed.

*Forward secrecy.* Since the one-way-AKE-fresh definition depends on the session output, and the protocol specifies the output, one-way-AKE-security does not necessarily imply forward secrecy. For example, in our `ntor` protocol in Sect. 6, the server's ephemeral and static public keys are in the same vector in the output, so the protocol has forward secrecy: if the static public key is (later) compromised, the session remains secure. By contrast, RSA key transport in TLS could be proven one-way-AKE-secure with only the server's static public key in the output, so it would not have forward secrecy (see Sect. 5.5). In general, a one-way-AKE-secure protocol gives forward secrecy if, whenever a party's long-term public key appears in an output vector, that output vector also includes an ephemeral public key from that party.

### 5.3 One-way anonymity

The goal of the adversary in the one-way anonymity security experiment is to guess which of two parties is participating in the key exchange. We use a proxy—run by the challenger—to relay communications to the party the adversary is trying to identify. The experiment is shown diagrammatically in Fig. 1. The adversary gives distinct party indices $i$ and $j$ to the challenger; the challenger randomly picks $i^*$ as one of these, activates $P_{i^*}$, and then relays messages between the adversary and $P_{i^*}$. The adversary's task is to guess $i^*$.

In the one-way anonymity security experiment, the adversary can issue the following queries to the challenger $\mathcal{C}$. The first two queries are for activation and communication during the test session:

– $\mathsf{Start}^{\mathcal{C}}(i, j, params, pid) \rightarrow msg'$: Abort if $i = j$. Otherwise, set $i^* \xleftarrow{\$} \{i, j\}$ and $(\Psi^*, msg') \leftarrow \mathsf{Send}^{P_{i^*}}(params, pid)$; return $msg'$. Only one call to $\mathsf{Start}^{\mathcal{C}}$ is allowed.
– $\mathsf{Send}^{\mathcal{C}}(msg) \rightarrow msg'$: Return $msg' \leftarrow \mathsf{Send}^{P_{i^*}}(\Psi^*, msg)$.

The remaining queries that can be issued by the adversary to the challenger $\mathcal{C}$ deal with the compromise of information related to the test session:

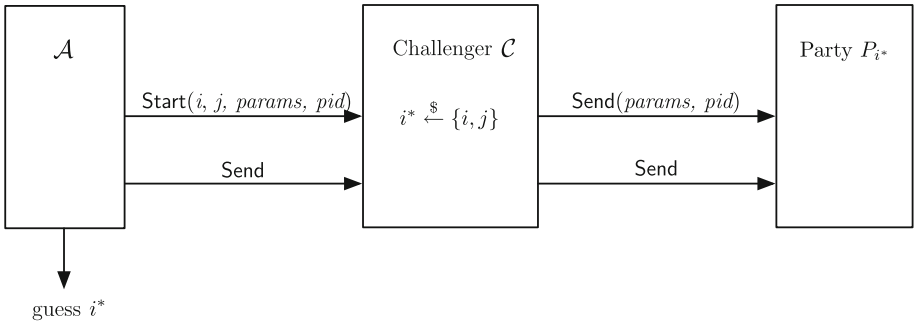


**Fig. 1** The challenger for the one-way anonymity security experiment

- $\mathsf{RevealNext}^{\mathcal{C}} \rightarrow X$: Return $\mathsf{RevealNext}^{P_{i^*}}$. Note that the responses to the $\mathsf{RevealNext}^{\mathcal{C}}$ and $\mathsf{RevealNext}^{P_{i^*}}$ queries are additionally modified so that no values produced by $\mathsf{RevealNext}^{\mathcal{C}}$ are used in sessions other than $\Psi^*$, and no values produced by the adversary's direct calls to $\mathsf{RevealNext}^{P_{i^*}}$ are used in session $\Psi^*$.
- $\mathsf{SessionKeyReveal}^{\mathcal{C}}() \rightarrow sk$: Return $sk \leftarrow \mathsf{SessionKeyReveal}^{P_{i^*}}(\Psi^*)$.
- $\mathsf{Partner}^{\mathcal{C}}(X) \rightarrow x$, where $X$ was a value returned by $\mathsf{Send}^{\mathcal{C}}$: Return $x \leftarrow \mathsf{Partner}^{P_{i^*}}(X)$.

**Definition 2** (*One-way anonymity*) Let $k$ be a security parameter and let $n \geq 1$. A protocol $\pi$ is said to be *one-way anonymous* if, for all probabilistic polynomial time (in $k$) adversaries $\mathcal{M}$, the advantage (over random guessing) that $\mathcal{M}$ wins the following experiment is negligible (in $k$):

- $\mathsf{Expt}^{\text{1w-anon}}_{\pi,k,n}(\mathcal{M})$:
  - Initialize $pubparams$.
  - Initialize parties $P_1, \ldots, P_n$.
  - Set $\hat{i} \leftarrow \mathcal{M}^{P_1,\ldots,P_n,\mathcal{C}}(pubparams)$.
  - Suppose that $\mathcal{M}$ made an $\mathsf{Start}^{\mathcal{C}}(i, j, params, pid)$ query which chose $i^*$. If $\hat{i} = i^*$, and $\mathcal{M}$'s queries satisfy the following constraints, then $\mathcal{M}$ wins, otherwise $\mathcal{M}$ loses.
    - No $\mathsf{SessionKeyReveal}(\Psi^*)$ query to $P_i$ or $P_j$.
    - No $\mathsf{Partner}(X)$ query to $P_i$ or $P_j$ for any value $X$ returned by $\mathcal{C}$.
    - No $\mathsf{Send}(\Psi^*, \cdot)$ query to $P_i$ or $P_j$.
    - Both $P_i$ and $P_j$ had exactly one certificate—the same certificate—for $pid$ during the run of the protocol for $\Psi^*$.

*Remark 1* The restrictions in Definition 2 are to prevent the adversary from trivially learning $P_{i^*}$ by querying $P_i$ and $P_j$ on information related to the target session. For example, if $i^* = i$, then $\mathsf{SessionKeyReveal}^{P_i}(\Psi^*)$ would return the real session key while $\mathsf{SessionKeyReveal}^{P_j}(\Psi^*)$ would return $\perp$ since $P_j$ has no session $\Psi^*$, and the adversary can then learn that $i^* = i$. We also ensure that both $P_i$ and $P_j$ hold exactly the same certificate for $pid$ during the entire test session.

However, we do allow the adversary to interact with the challenger in much the same way as a normal party. The adversary can compromise values for the test session: the main restriction is that it must do so via queries to the challenger, not queries to the actual parties. In order to further prevent trivial attacks, we must ensure that any values used in the test session are not used in other sessions at the party, and this is accounted for in the definition of $\mathsf{RevealNext}^{\mathcal{C}}$.

#### 5.3.1 Related notion: External anonymity

Definition 2 hides the identity of one party (say the initiator of a Tor connection) from that party's peer (the Tor server). This *internal anonymity* property is different from *external anonymity* where the identity of the communicating parties is hidden from most parties on the network but become available to the session participants by the end of the protocol. This property has been considered previously [1,7]; it can be achieved via encryption of the identities of communicating parties.

#### 5.3.2 Related notion: Two-way anonymity

Definition 2 can also be extended to provide anonymity for both peers from *each other*. However, as we argued earlier, authentication bounds the level of security. With two-way anonymity, no authentication is provided, so secrecy is not well-motivated. Thus, for one-way AKE protocols the one-way anonymity definition suffices.

#### 5.3.3 Related notion: Unlinkability

*Unlinkability* is the property that an attacker cannot determine whether two items of interest are related or not [32, §4]. In the context of anonymous key exchange, the two items of interest could be two key exchange sessions, and the relation the attacker aims to determine is whether the two sessions are with the same anonymous party or with two different anonymous parties.

With a suitable formalization of unlinkability of anonymous key exchange sessions, it can be shown that unlinkablity is equivalent to one-way anonymity as defined in Sect. 5.3. We present a formalization and argument for this in Appendix A.

#### 5.3.4 Relation notion: Deniability

*Deniability* is the property that a party's participation in a protocol cannot be proven conclusively to a judge: while the session owner may be convinced of her peer's identity, the transcript cannot convince someone else of the same fact. For example, in the context of authenticated key agreement protocols [11], a protocol in which authentication is demonstrated using a digital signature, deniability is in general not satisfied, but a protocol in which authentication is demonstrated by correctly decrypting a public-key ciphertext may achieve deniability.

Deniability is not the same as anonymity. While in a deniable protocol it cannot be proven that a session involves a particular party, typical sessions between honest parties may still leak identifying information about the parties involved. In contrast, protocols with anonymity properties ensure stronger privacy properties in that no information about the parties involved is leaked.

### 5.4 One-way AKE protocols

Given the model, it is worth considering which protocols satisfy the security model we propose. Natural candidates are the KAS1 family of protocols in SP800-56B [29] and the so called $C(1, 1)$ family of protocols in SP800-56A [28]. In both families of protocols the responder provides a static key pair and possibly a nonce—a one-time random public string. One easily verifies that these protocols do not provide forward secrecy. If the static private

key of the server is compromised, the client, who contributed the ephemeral key pair, loses secrecy. We are interested in the assurances that would satisfy the needs of the Tor network, where forward secrecy is an important attribute. In the next section we propose and analyze such a protocol, but it is also of interest to see if forward secrecy is the only drawback of the protocols suggested in SP800-56A and SP800-56B.

### 5.5 Analysis of TLS

The Transport Layer Security (TLS) protocol [13] is most commonly used to provide confidentiality and authentication on the web, and in that setting it is typical that only server authentication is used. It is natural, then, to consider whether TLS satisfies the one-way AKE and anonymity security properties. TLS supports a wide variety of cryptographic algorithms in the form of *ciphersuites*; we will focus on two common ciphersuites, namely RSA key transport (TLS_RSA_WITH_AES_128_CBC_SHA) and signed ephemeral Diffie-Hellman (TLS_DHE_RSA_WITH_AES_128_CBC_SHA).

*One-way-AKE security.* An essential characteristic of one-way-AKE security (Definition 1) is that, if a party uses multiple keys in a session, then the session should remain secure if all but one of those keys is compromised.

In RSA key transport in TLS, the client picks a random pre-master secret and sends it to the server encrypted under the server's long-term public key; the server has no ephemeral key. Hence, provided the server's long-term key is not revealed, the session remains secure, and TLS could theoretically be proven secure in the formalism of Definition 1. However, as mentioned earlier, this does not provide forward secrecy.

In signed ephemeral Diffie-Hellman key exchange in TLS, the client and server use plain Diffie-Hellman key exchange to establish a shared secret, and the server signs its ephemeral Diffie-Hellman key using its long-term signing key. If the server's ephemeral key is revealed, the session key is no longer secure, and thus this ciphersuite cannot satisfy our one-way-AKE security definition.

*Anonymity.* The core cryptographic components of TLS — basic RSA key transport or the basic ephemeral Diffie-Hellman — appear at first glance to provide anonymity for clients in the fashion of Definition 2. However, TLS as a protocol consists of much more than the core cryptographic components; it includes protocol negotiation (the ClientHello and ServerHello messages), session reuse, and other values in various messages. We have identified several parts of the protocol that have the potential to break client anonymity (even when used over an anonymizing network protocol such as Tor):

- In the ClientHello message, the client indicates its supported TLS versions, ciphersuites, algorithms, and extensions. These values may leak information about the client's configuration.
- In the client_random value in the ClientHello message, the client provides its computer's current time in seconds (gmt_unix_time). As the clocks of various client computers may not all be exactly synchronized on a universal time, this value may allow a server to link client sessions.
- The session identifier of a new session is specified by the server in the session_id field of the ServerHello message. In particular, the client has no input into the session identifier. As a result, it is possible for a malicious server to reuse the same session identifier in multiple sessions. While the TLS specification does not explicitly require a client to abort

if it receives the same session identifier from a server twice, many implementations do abort in that case.[6]

## 6 Proposed Protocol `ntor`

In this section we propose and analyze a one-way authenticated key exchange protocol that provides one-way anonymity. Our protocol is suitable for use in a variety of settings, including the Tor anonymity network [37]. We describe the protocol as being between a "client" and a "server" but the protocol could equally be considered as being among parties who take on multiple roles as both "initiators" and "responders", albeit with independent keys in their separate roles.

**Definition 3** Let $k$ be a security parameter. Let $\mathcal{G}$ be a cyclic group of prime order $q$ generated by an element $g$, and let $\mathcal{G}^*$ be the set of non-identity elements of $\mathcal{G}$. Let $\mathrm{H}_{sid}, \mathrm{H}_{mac}$ be hash functions with range $\{0,1\}^k$ and let $\mathrm{H}$ be a hash function with range $\{0,1\}^k \times \{0,1\}^k$. The `ntor` protocol proceeds as follows:

– When $\hat{B}$ is initialized as a server:

1. Set $b \xleftarrow{\$} \{1, \ldots, q-1\}$ and set $B \leftarrow g^b$.
2. Set $(b, B)$ as $\hat{B}$'s static key pair.
3. Set $cert_B = (\hat{B}, B)$ as $\hat{B}$'s certificate.

– When $\hat{A}$ is initialized as a client:

4. Obtain an authentic copy of $\hat{B}$'s certificate.

– When $\hat{A}$ receives the message $(params, pid) = (\text{("new session"}, \texttt{ntor}), \hat{B})$:

5. Verify that $\hat{A}$ holds an authenticated certificate $cert_B = (\hat{B}, B)$.
6. Obtain an unused ephemeral key pair $(x, X \leftarrow g^x)$; set session id $\Psi_a \leftarrow \mathrm{H}_{sid}(X)$.
7. Set $M^{\hat{A}}_{\mathbf{state}}[\Psi_a] \leftarrow (\texttt{ntor}, \hat{B}, x, X)$.
8. Return session identifier $\Psi_a$ and outgoing message $msg' \leftarrow (\texttt{ntor}, \hat{B}, X)$.

– When $\hat{B}$ receives the message $msg = (\texttt{ntor}, \hat{B}, X)$:

9. Verify $X \in \mathcal{G}^*$.
10. Obtain an unused ephemeral key pair $(y, Y \leftarrow g^y)$; set session id $\Psi_b \leftarrow \mathrm{H}_{sid}(Y)$.
11. Compute $(sk', sk) = \mathrm{H}(X^y, X^b, \hat{B}, X, Y, \texttt{ntor})$.
12. Compute $t_B = \mathrm{H}_{mac}(sk', \hat{B}, Y, X, \texttt{ntor}, \text{"server"})$.
13. Return session identifier $\Psi_b$ and outgoing message $msg' \leftarrow (\texttt{ntor}, Y, t_B)$.
14. Complete $\Psi_b$ by deleting $y$ and outputting $(sk, \circledast, (\mathbf{v}_0, \mathbf{v}_1))$, where $\mathbf{v}_0 = (X)$ and $\mathbf{v}_1 = (Y, B)$.

– When $\hat{A}$ receives the message $msg = (\texttt{ntor}, Y, t_B)$ for session identifier $\Psi_a$:

15. Verify session state $M^{\hat{A}}_{\mathbf{state}}[\Psi_a]$ exists.
16. Retrieve $\hat{B}$, $x$, and $X$ from $M^{\hat{A}}_{\mathbf{state}}[\Psi_a]$.
17. Verify $Y \in \mathcal{G}^*$.
18. Compute $(sk', sk) = \mathrm{H}(Y^x, B^x, \hat{B}, X, Y, \texttt{ntor})$.

[6] For example, in OpenSSL [30], the routine ssl_get_new_session in the file ssl/ssl_sess.c aborts if the session_id exists in the cache.

19. Verify $t_B = \mathrm{H}_{mac}(sk', \hat{B}, Y, X, \texttt{ntor}, \text{“server”})$.
20. Complete $\Psi_a$ by deleting $M^{\hat{A}}_{\textbf{state}}[\Psi_a]$ and outputting $(sk, \hat{B}, (\mathbf{v}_0, \mathbf{v}_1))$, where $\mathbf{v}_0 = (X)$ and $\mathbf{v}_1 = (Y, B)$.

If any verification fails, the party erases *all* session-specific information and aborts the session.

### 6.1 Security

Security of the proposed protocol is established via the following theorem.

**Theorem 1** *Assume* H *and* $\mathrm{H}_{mac}$ *are random oracles and* $\mathrm{H}_{sid}$ *is collision-resistant.*[7] *If* $\mathcal{G}$ *is a group where the gap Diffie-Hellman assumption holds, then the* ntor *protocol in Definition 3 is a one-way-AKE-secure protocol.*

*Argument outline:* The argument is a reduction $\mathcal{S}$ that takes an CDH instance $(U, V)$; $\mathcal{S}$ has access to both a DDH oracle and $\mathcal{M}$. The algorithm $\mathcal{M}$ can distinguish a fresh session key from a randomly chosen key. Since $\mathcal{M}$ is polynomially bounded, $\mathcal{S}$ has a non-negligible chance to guess in advance the test session and which public keys will be involved in the test session. By embedding into the test session's public keys the values $(U, V)$ and observing the random oracle queries the algorithm $\mathcal{S}$ can extract the solution to the CDH instance. It is important to have an authenticated peer in the test session as it allows the challenger to embed the CDH instance within a public key that has not been manipulated by the adversary. The remaining challenge is to provide consistent random oracle queries and session key reveal queries in case $\mathcal{S}$ does not posses the corresponding private keys. The consistency is achieved by utilizing the DDH oracle. The details are provided in Appendix B.

### 6.2 Anonymity

**Theorem 2** *The* ntor *protocol in Definition 3 is a one-way anonymous protocol according to Definition 2.*

*Argument:* We proceed by introducing another experiment, which the adversary cannot win more often than random guessing. In this new experiment, the choice of $i^*$ will be uncoupled from the behaviour of the rest of the system. We then show that no adversary can distinguish this new experiment from the original experiment, thereby showing the anonymity of the protocol.

Let $\mathsf{Expt}^{\text{1w-anon}'}_{\texttt{ntor},k,n}(\mathcal{M})$ be the same experiment as $\mathsf{Expt}^{\text{1w-anon}}_{\texttt{ntor},k,n}(\mathcal{M})$, except that $\mathcal{C}$ uses the following new oracles for the 1w-anon$'$ experiment instead of the original ones from the 1w-anon experiment:

– $\mathsf{Start}'(i, j, params, pid = \hat{B}) \rightarrow msg'$:

1. Abort if $i = j$.
2. Set $i^* \xleftarrow{\$} \{i, j\}$.
3. Obtain an unused ephemeral key pair $(x^*, X^*)$, consistent with the $\mathsf{RevealNext}'$ query.
4. Set $\Psi^* \leftarrow \mathrm{H}_{sid}(X^*)$.
5. Set $\hat{B}^* \leftarrow \hat{B}$ and store $M^*_{\textbf{state}} \leftarrow (\texttt{ntor}, \hat{B}^*, x^*, X^*)$.
6. Return session identifier $\Psi^*$ and outgoing message $msg' \leftarrow (\texttt{ntor}, \hat{B}^*, X^*)$.

[7] For example, $\mathrm{H}_{sid}$ may in fact be the identity function.

– $\mathsf{Send}'(msg = (\texttt{ntor}, Y, t_b))$:

  1. Verify session state $M^*_{\mathbf{state}}$ exists.
  2. Retrieve $\hat{B}^*$, $x^*$, and $X^*$ from $M^*_{\mathbf{state}}$.
  3. Verify $Y \in \mathcal{G}^*$.
  4. Compute $(sk', sk) \leftarrow \mathrm{H}(Y^{x^*}, B^{x^*}, \hat{B}^*, X^*, Y, \texttt{ntor})$.
  5. Verify $t_B = \mathrm{H}_{mac}(sk', \hat{B}, Y, X, \texttt{ntor}, \text{“server”})$.
  6. Complete by deleting $M^*_{\mathbf{state}}$ and outputting $(sk, \hat{B}^*, \{(X^*), (Y, B)\})$.

– $\mathsf{SessionKeyReveal}'() \rightarrow sk$: If the test session has completed, return $sk$.
– $\mathsf{Partner}'(X^*)$: Return $M^*_{\mathbf{state}}$, $out^*$, or $x^*$, respectively.
– $\mathsf{RevealNext}'(eph, n) \rightarrow (X_1, \ldots, X_n)$: Return $n$ values generated by $F_{eph}$, consistent with previous $\mathsf{RevealNext}'$ and $\mathsf{Send}'$ queries.

Since all messages computed in the 1w-anon′ experiment are independent of the choice of $i^*$, $\mathcal{M}$ has no advantage:

$$\Pr\left(\mathsf{Expt}^{\text{1w-anon}'}_{\texttt{ntor},k,n}(\mathcal{M}) = \text{win}\right) = \frac{1}{2}. \tag{1}$$

We now show that no adversary can distinguish the two experiments.

The distribution of messages returned by the challenger in the 1w-anon′ experiment is identical to messages returned in the 1w-anon experiment. Moreover, the messages from all parties except $P_i$ and $P_j$ are unchanged. For messages from $P_i$ and $P_j$, all queries return identical distributions of messages in the 1w-anon′ experiment as in the 1w-anon experiment.

We note that any other queries that might reveal information about which of $P_i$ or $P_j$ has participated in the test session are prohibited by Definition 2. For example, the adversary is prohibited from trying to determine if $P_i$ holds a session key related to the target session using the query $\mathsf{SessionKeyReveal}(\Psi^*)$ to $P_i$.

Thus, the two experiments are indistinguishable from an adversary's perspective:

$$\Pr\left(\mathsf{Expt}^{\text{1w-anon}}_{\texttt{ntor},k,n}(\mathcal{M}) = \text{win}\right) = \Pr\left(\mathsf{Expt}^{\text{1w-anon}'}_{\texttt{ntor},k,n}(\mathcal{M}) = \text{win}\right). \tag{2}$$

Combining equations (1) and (2) shows that the ntor protocol is one-way anonymous. □

### 6.3 Efficiency

Table 1 presents the efficiency in term of group exponentiations of relevant key agreement protocols. The Diffie-Hellman protocol [14] is the basic protocol on which most other protocols in the literature are built upon. In the table we refer to the ephemeral-ephemeral variant that succumbs to man-in-the-middle attacks, but is a good benchmark for efficiency. We also include the "fourth protocol" of Øverlier and Syverson [31], denoted by ØS, which was proposed as an efficient Tor candidate that provides one-way authentication. The MQV [23] and the UM [5] key agreement protocols were initially devised to provide two parties holding certified static keys with a session key. Variants where parties do not provide static keys are included in the NIST SP800-56A [28] standard. In case a party does not contribute a static key, the ephemeral key substitutes for the static key in all computations. We emphasize that SP800-56A does not allow the scenario where the initiator contributes only an ephemeral key and the responder contributes both static and ephemeral keys. In this comparison we use the variants where both participants contribute both static and ephemeral key pairs. Variants of both protocols have been shown secure; however, the UM protocol achieves security in a restricted model, whereas the MQV variant (HMQV [21]) proven secure has a highly

**Table 1** Efficiency in terms of group exponentiations

| Protocol | Efficiency (client) | | Efficiency (server) | | Authentication | Security |
|---|---|---|---|---|---|---|
| | Off-line | On-line | Off-line | On-line | | |
| DH | 1 | 1 | 1 | 1 | None | Insecure |
| ØS | 1 | 1 | 1 | 1 | One-way | Insecure |
| MQV | 1 | 1.17 (1.5) | 1 | 1.17 (1.5) | Mutual | Non-tight |
| UM | 1 | 2 | 1 | 2 | Mutual | Limited |
| Ours | 1 | 2 | 1 | 1.33 (2) | One-way | Tight |

non-tight security argument. By "tight" in the context of key agreement security reductions, we mean that the only significant factor between the difficulty of breaking the key agreement protocol and the difficulty of solving the underlying function is the factor that comes from guessing the correct test session. This factor is present in all known security reductions of key agreement protocols.

In the table all protocols except ours are balanced in the sense that the operations performed by both the client and the server have the same cost. The values in the brackets are naive counts and public key validation is omitted. MQV benefits from Shamir's trick for simultaneous multiple exponentiation [25, Algorithm 14.88]. In our protocol the server needs to compute $X^b$ and $X^y$. Since the base is the same, squarings in the square-and-multiply algorithm can be parallelized [27] reducing the computational cost to 1.33 exponentiations. Further improvements such the "Exponent Combination Method" [27, §2.3] can be applied to the computation of the server. However such algorithms further increase the complexity of the computations and the improvements are not always clear cut. Note that the off-line stage can also benefit from the Exponent Combination Method. We do not consider amortized costs since they are applicable to all protocols and will not affect any comparison.

On the client side our proposal is at least as efficient as the UM protocol. Since the exponent in $B^x$ and $Y^x$ is repeated there may be room for improvement. However such improvements will provide little contribution since clients are expected to run relatively few sessions compared to the server. On the server side our protocol performs almost as efficiently as the MQV protocol. While improved computation techniques or protocols may be feasible, the narrow gap with the DH protocol indicates that our protocol is very efficient on the server side without a compromise in security.

## 7 Conclusions and future work

This work provides a general approach to one-way authenticated key establishment. We motivated the need for a formal approach by analyzing and finding a major flaw in an existing protocol that aims to achieve one-way authentication. Subsequently, we identified the practical motivations for authentication and secrecy assurances of parties engaging in one-way AKE protocols. Based on our discussion we proposed a suitable model that covers those goals and offered an efficient protocol that formally meets the proposed security definition.

In our discussion we related our security definition not only with usual mutually authenticated key establishment protocols but also with public key encryption. It is of interest

to formally investigate the relation between public key encryption and our one-way AKE security definitions. Furthermore, identifying and formalizing other goals such as aggregate security or exclusivity, and looking for the relation with our definitions is another worthy objective.

**Acknowledgments** The authors gratefully acknowledge helpful discussions with Alfred Menezes and Paul Syverson. I. Goldberg acknowledges the financial support of the Natural Sciences and Engineering Research Council (NSERC) of Canada and Mprime.

## A Equivalence of unlinkability and one-way anonymity

In this section we provide a formalization of the unlinkability security property from Sect. 5.3.3 and show that it is equivalent to one-way anonymity (Definition 2).

The goal of the adversary in the unlinkability security experiment is to determine whether two sessions are being run at the same party or at different parties. We use a proxy—run by the challenger—to relay communications for the sessions the adversary is trying to link. The experiment is shown diagrammatically in Fig. 2. The adversary gives distinct party indices $i$ and $j$ to the challenger and then participates in two sessions via the challenger. The challenger randomly picks a bit $b^*$; if $b^* = 0$, then the challenger runs one session at party $P_i$ the other session at party $P_j$; if $b^* = 1$, then the challenger runs both sessions at party $P_i$. The adversary's task is to guess $b^*$.

### A.1 Security experiment

In the unlinkability security experiment, the adversary can issue the following queries to the challenger $\mathcal{C}$. The first three queries are for activation and communication during the test session:

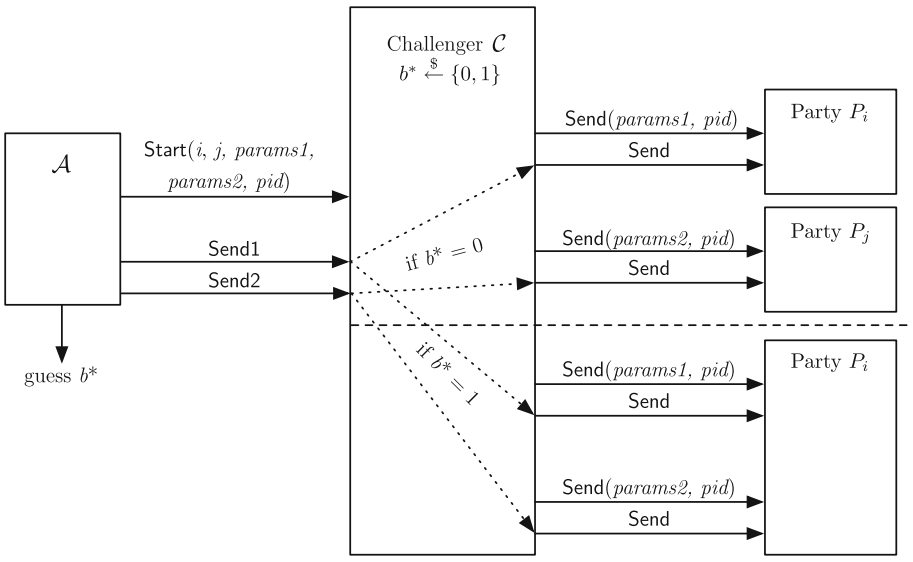


**Fig. 2** The challenger for the unlinkability security experiment

1. $\mathsf{Start}^{\mathcal{C}}(i, j, params1, params2, pid) \rightarrow (msg'_1, msg'_2)$: Abort if $i = j$. Otherwise, set $b^* \xleftarrow{\$} \{0, 1\}$. Set $i^* \leftarrow i$. If $b^* = 0$ then set $j^* \leftarrow j$, otherwise set $j^* \leftarrow i$. Set $(\Psi_1^*, msg'_1) \leftarrow \mathsf{Send}^{P_{i^*}}(params1, pid)$ and $(\Psi_2^*, msg'_2) \leftarrow \mathsf{Send}^{P_{j^*}}(params2, pid)$. Return $(msg'_1, msg'_2)$. Only one call to $\mathsf{Start}^{\mathcal{C}}$ is allowed.
2. $\mathsf{Send1}^{\mathcal{C}}(msg) \rightarrow msg'$: Return $msg' \leftarrow \mathsf{Send}^{P_{i^*}}(\Psi_1^*, msg)$.
3. $\mathsf{Send2}^{\mathcal{C}}(msg) \rightarrow msg'$: Return $msg' \leftarrow \mathsf{Send}^{P_{j^*}}(\Psi_2^*, msg)$.

The remaining queries that can be issued to the challenger—$\mathsf{RevealNext1}^{\mathcal{C}}$, $\mathsf{RevealNext2}^{\mathcal{C}}$, $\mathsf{SessionKeyReveal1}^{\mathcal{C}}$, $\mathsf{SessionKeyReveal2}^{\mathcal{C}}$, $\mathsf{Partner1}^{\mathcal{C}}$, and $\mathsf{Partner2}^{\mathcal{C}}$—deal with the compromise of information related to the test sessions, and are analogous to the queries in the one-way anonymity experiment (Sect. 5.3), where queries labelled 1 are relayed to party $P_{i^*}$ and parties labelled 2 are relayed to party $P_{j^*}$.

**Definition 4 (Unlinkability)** Let $k$ be a security parameter and let $n \geq 1$. A protocol $\pi$ is said to be *unlinkable* if, for all probabilistic polynomial time (in $k$) adversaries $\mathcal{M}$, the advantage (over random guessing) that $\mathcal{M}$ wins the following experiment is negligible (in $k$):

– $\mathsf{Expt}^{\text{unlink}}_{\pi,k,n}(\mathcal{M})$:
  – Initialize $pubparams$.
  – Initialize parties $P_1, \ldots, P_n$.
  – Set $\hat{b} \leftarrow \mathcal{M}^{P_1,\ldots,P_n,\mathcal{C}}(pubparams)$.
  – Suppose that $\mathcal{M}$ makes a $\mathsf{Start}^{\mathcal{C}}(i, j, params1, params2, pid)$ query which chose $b^*$. If $\hat{b} = b^*$, and $\mathcal{M}$'s queries satisfy the following constraints, then $\mathcal{M}$ wins, otherwise $\mathcal{M}$ loses.
    - No $\mathsf{SessionKeyReveal}(\Psi_1^*)$ or $\mathsf{SessionKeyReveal}(\Psi_2^*)$ query to $P_i$ or $P_j$.
    - No $\mathsf{Partner}(X)$ query to $P_i$ or $P_j$ for any value $X$ returned by $\mathcal{C}$.
    - No $\mathsf{Send}(\Psi_1^*, \cdot)$ or $\mathsf{Send}(\Psi_2^*, \cdot)$ query to $P_i$ or $P_j$.
    - Both $P_i$ and $P_j$ had exactly one certificate—the same certificate—for $pid$ during both entire test sessions (that is, from when $\mathsf{Start}^{\mathcal{C}}$ was called until both $\mathsf{Send1}^{\mathcal{C}}$ and $\mathsf{Send2}^{\mathcal{C}}$ resulted in accepted sessions).

## A.2 Equivalence

**Theorem 3** *A protocol $\pi$ is unlinkable if and only if it is one-way anonymous.*

*Argument:* The proof proceeds using a bilateral simulation. First, we show that an adversary against the unlinkability experiment can be used to break the one-way anonymity experiment, and then we show the reverse, yielding the equivalence of the two properties.

*[One-way anonymity $\Longrightarrow$ unlinkability.]* Let $\mathcal{M}^u$ be an adversary against the unlinkability experiment. We will construct a simulator $\mathcal{S}$ that responds to $\mathcal{M}^u$'s queries but is itself an adversary against the one-way anonymity experiment. Let $\mathcal{C}^a$ be the challenger for the one-way anonymity experiment. The basic idea is that $\mathcal{S}$ will run one session with party $P_i$ and the other session with $\mathcal{C}^a$, and $\mathcal{C}^a$ may be running the session either with $P_i$ or $P_j$.

When $\mathcal{M}^u$ sends $\mathcal{S}$ a $\mathsf{Start}(i, j, params1, params2, pid)$ query, the simulator $\mathcal{S}$ sends $P_i$ a $\mathsf{Send}(params1, pid)$ query and receives $msg'_1$; $\mathcal{S}$ also sends $\mathcal{C}^a$ a $\mathsf{Start}(i, j, params2, pid)$ query and receives $msg'_2$. $\mathcal{S}$ returns $(msg'_1, msg'_2)$ to $\mathcal{M}^u$. When $\mathcal{M}^u$ sends $\mathcal{S}$ a $\mathsf{Send1}$ query, $\mathcal{S}$ relays that query to $P_i$ and returns the responses, but for $\mathsf{Send2}$ queries, $\mathcal{S}$ relays those queries to $\mathcal{C}^a$. $\mathsf{RevealNext}$, $\mathsf{SessionKeyReveal}$, and $\mathsf{Partner}$ queries made against $\mathcal{S}$

are handled analogously. All other queries that $\mathcal{M}^u$ makes against parties are relayed directly to those parties.

It is easy to see that the distributions of all messages returned to $\mathcal{M}^u$ are identical to those in the original unlinkability experiment. Moreover, provided that $\mathcal{M}^u$ makes no prohibited (according to Definition 4) queries to $\mathcal{S}$, then $\mathcal{S}$ will make no prohibited (according to Definition 2) queries to $\mathcal{C}^a$.

If $\mathcal{C}^a$ is using $i^* = i$, then the simulated experiment is behaving exactly as if it were the unlinkability experiment with $b^* = 1$, and if $\mathcal{C}^a$ is using $i^* = j$, then the simulated experiment is behaving exactly as if it were the unlinkability experiment with $b^* = 0$. Hence, if $i^*$ is the output of $\mathcal{M}^u$, then $\mathcal{S}^u$ returns $b^* \leftarrow 1$ if $i^* = i$, and $b^* \leftarrow 0$ otherwise. If $\mathcal{M}^u$ is an adversary against the unlinkability experiment with non-negligible probability, then $\mathcal{S}$ is an adversary against the one-way anonymity experiment with the same non-negligible probability.

*[Unlinkability $\implies$ one-way anonymity.]* Let $\mathcal{M}^a$ be an adversary against the one-way anonymity experiment. We will construct a simulator $\mathcal{S}$ that responds to $\mathcal{M}^a$'s queries but is itself an adversary against the unlinkability experiment. Let $\mathcal{C}^u$ be the challenger for the unlinkability experiment. The basic idea is that $\mathcal{S}$ will run the test session as one of the two sessions in the unlinkability experiment with $\mathcal{C}^u$, and $\mathcal{C}^u$ may be running that session either as $P_i$ or $P_j$. The simulator $\mathcal{S}$ then uses $\mathcal{M}^a$'s guess of whether the one-way anonymity session is run at $P_i$ or $P_j$ as $\mathcal{S}$'s guess against the unlinkability challenger $\mathcal{C}^u$ of whether the two sessions are run at the same party or at different parties.

When $\mathcal{M}^a$ sends a $\mathsf{Start}(i, j, params, pid)$ query to $\mathcal{S}$, the simulator $\mathcal{S}$ sends $\mathcal{C}^u$ a $\mathsf{Start}(i, j, params1, params, pid)$ query, for a random $params1$ chosen according to the protocol, receives $(msg'_1, msg'_2)$, and returns $msg' \leftarrow msg'_2$ to $\mathcal{M}^a$. When $\mathcal{M}^a$ sends $\mathcal{S}$ a $\mathsf{Send}$ query, $\mathcal{S}$ relays that message to $\mathcal{C}^u$ via a $\mathsf{Send2}$ query and returns the response. $\mathsf{RevealNext}$, $\mathsf{SessionKeyReveal}$, and $\mathsf{Partner}$ queries made against $\mathcal{S}$ are handled analogously. All other queries that $\mathcal{M}^u$ makes against parties are relayed directly to those parties.

It is easy to see that the distributions of all messages returned to $\mathcal{M}^a$ are identical to those in the original one-way anonymity experiment. Moreover, provided that $\mathcal{M}^a$ makes no prohibited (according to Definition 2) queries to $\mathcal{S}$, then $\mathcal{S}$ will make no prohibited (according to Definition 4) queries to $\mathcal{C}^u$.

If $\mathcal{C}^u$ is using $b^* = 0$, then the simulated experiment is behaving exactly as if it were the one-way anonymity experiment with $i^* = j$, and if $\mathcal{C}^a$ is using $b^* = 1$, then the simulated experiment is behaving exactly as if it were the one-way anonymity experiment with $i^* = i$. Hence, if $b^*$ is the output of $\mathcal{M}^a$, then $\mathcal{S}^u$ returns $i^* \leftarrow i$ if $b^* = 1$, and $i^* \leftarrow j$ otherwise. If $\mathcal{M}^a$ is an adversary against the one-way anonymity experiment with non-negligible probability, then $\mathcal{S}$ is an adversary against the unlinkability experiment with the same non-negligible probability. □

## B Security argument

*Argument:* It is straightforward to verify that the protocol is correct, i.e., that if messages are delivered faithfully the sessions produce the same session key.

We verify that the session key is indistinguishable from a randomly chosen session key. Suppose the test session is $\Psi_A$ with output $(sk, \hat{B}, \{(X), (Y, B)\})$. We emphasize that the list of identities in the public output cannot be empty since without an authenticated peer anyone—in particular the adversary—can be the session peer.

In the random oracle model, the adversary can distinguish the session key from a randomly chosen session key only if the adversary queries the oracle with the same input as the test session owner. We first argue that no two sessions with different public outputs have the same input to the key derivation.

On one hand for any party $\hat{B}$ that contributes a certified public key $\hat{B}$ also contributes an ephemeral public key chosen on a per-session basis. Therefore by including the identifier of the certificate along with the static and ephemeral public keys, $\hat{B}$ is assured that the key derivation material of all sessions for which $\hat{B}$ is a partner are different from each other.

On the other hand for any party $\hat{A}$ the ephemeral public key $X$ contributed by $\hat{A}$ implies that except with negligible probability no other honest party will input $X$ to the key derivation function. Including the identifier $\hat{B}$ assures that only the intended peer $\hat{B}$ will input it to the key derivation function. The string `ntor` assures $\hat{A}$ that $\hat{B}$ is aware about the key agreement protocol for which the static public key $B$ is used.

Since the actions of malicious parties cannot be predicted, the above argument implicitly assumes that parties are honest. Thus for honest parties the public information input to the key derivation function suffices to ensure that different sessions have different inputs. Therefore exposing any session key owned by an honest party reveals no information about the test session key. To show security we will argue that no malicious entity can compute the entire input to the key derivation function unless the entity becomes a partner to at least one vector from the public output.

If the adversary $\mathcal{M}$ is partner to $X$ then the session key is compromised. So in the remainder we assume that $\mathcal{M}$ is not a partner to $X$ and the test session has output

$$(sk, \hat{B}, \{(X), (Y, B)\})$$

where $sk$ is not known to $\mathcal{M}$. Since the key derivation function is modeled as a random oracle the only way $\mathcal{M}$ can distinguish the session key from a randomly chosen key is to query `H` with

$$(\mathrm{CDH}(X, Y), \mathrm{CDH}(X, B), \hat{B}, Y, X, \texttt{ntor}).$$

To show that breaking protocol's security is equivalent to solving a CDH instance $(U, V)$ we outline an algorithm $\mathcal{S}$ that solves the instance. The algorithm is given a black box access to $\mathcal{M}$ and simulates its environment. Since the number of activated sessions is polynomially bounded, with non-negligible probability $\mathcal{S}$ can correctly guess which public key $X$ will be the test sessions' outgoing ephemeral public key. In the remainder of this argument we assume that this event occurred. In the simulation $\mathcal{S}$ sets $X = U$; since $\mathcal{M}$ is not partner of $X$, the adversary cannot distinguish this deviation from an honest protocol simulation. A successful adversary is partner to at most one of $Y$ and $B$.

Assume $\mathcal{M}$ is not a partner of $Y$. Since the number of ephemeral key pairs is polynomially bounded, with non-negligible probability $\mathcal{S}$ correctly anticipates and sets $Y = V$. Since $\mathcal{M}$ does not obtain the session key from the matching session if one exists[8], the adversary must query `H` with

$$(\sigma_1 = \mathrm{CDH}(U, V), \sigma_2 = \mathrm{CDH}(U, B), \hat{B}, V, U, \texttt{ntor}).$$

From all queries of this form $\mathcal{S}$ extracts the correct value $\sigma_1$ using the DDH oracle.

[8] The adversary may choose to use an ephemeral key from a non-matching session to supply the test session with an incoming ephemeral public key.

Assume $\mathcal{M}$ does not partner with $B$. Since the number of static key pairs is polynomially bounded, with non-negligible probability $\mathcal{S}$ correctly anticipates and sets $B = V$. In this case $\mathcal{S}$ has to simulate session executions without knowledge of the corresponding private key. In particular, the simulation may fail due to inconsistent H queries. To address this problem $\mathcal{S}$ uses the DDH oracle to respond consistently to H and SessionKeyReveal queries for sessions that involve $V$. In particular for H queries that involve $B$, $\mathcal{S}$ first verifies using the DDH oracle that the shared secrets are computed honestly before responding with the session key. As before $\mathcal{M}$ does not obtain the session key from the matching session, if one exists; therefore, the adversary must query H with

$$(\sigma_1 = \mathrm{CDH}(U, Y), \sigma_2 = \mathrm{CDH}(U, V), \hat{B}, Y, U, \mathtt{ntor}).$$

From all queries of this form $\mathcal{S}$ extracts the correct value $\sigma_2$ using the DDH oracle.

During the simulation, for each adversary query $\mathcal{S}$ performs a polynomially bounded number of steps and therefore the running time of the reduction is polynomially bounded. Therefore if $\mathcal{M}$ is successful with non-negligible probability $\mathcal{S}$ produces a solution to the CDH instance with non-negligible probability contradicting the hardness of the GDH assumption in the underlying group.

## References


1. Aiello W., Bellovin S.M., Blaze M., Canetti R., Ioannidis J., Keromytis A.D., Reingold O.: Just Fast Keying: key agreement in a hostile Internet. ACM Trans. Inform. Syst. Secur. **7**(2), 1–30 (2004). doi:10.1145/996943.996946.
2. Bellare M., Rogaway P.: Entity authentication and key distribution. In: Stinson D.R. (ed.) Advances in Cryptology—Proc. CRYPTO '93, LNCS, vol. 773, pp. 232–249. Springer (1993). doi:10.1007/3-540-48329-2_21.
3. Bellare M., Pointcheval D., Rogaway P.: Authenticated key exchange secure against dictionary attacks. In: Preneel B. (ed.) Advances in Cryptology—Proc. EUROCRYPT 2000, LNCS, vol. 1807, pp. 139–155. Springer (2000). doi:10.1007/3-540-45539-6_11.
4. Bellovin S.M., Merritt M.: Encrypted key exchange: password-based protocols secure against dictionary attacks. In: Proceedings of the 1992 IEEE Computer Society Conference on Research in Security and Privacy. IEEE (1992). doi:10.1109/RISP.1992.213269.
5. Blake-Wilson S., Johnson D., Menezes A.: Key agreement protocols and their security analysis. In: Darnell M. (ed.) Cryptography and Coding—6th IMA International Conference, LNCS, vol. 1355. Springer (1997). doi:10.1007/BFb0024447.
6. Canetti R., Krawczyk H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann B. (ed.) Advances in Cryptology—Proc. EUROCRYPT 2001, LNCS, vol. 2045, pp. 453–474. Springer (2001). doi:10.1007/3-540-44987-6_28
7. Canetti R., Krawczyk H.: Security analysis of IKE's signature based key-exchange protocol. In: Yung M. (ed.) Advances in Cryptology—Proc. CRYPTO 2002, LNCS, vol. 2442, pp. 27–52. Springer (2002). doi:10.1007/3-540-45708-9_10. Full version available as http://eprint.iacr.org/2002/120.
8. Cheng Z., Chen L., Comley R., Tang Q.: Identity-based key agreement with unilateral identity privacy using pairings. In: Chen K., Deng R., Lai X., Zhou J. (eds.) Proc. Information Security Practice and Experience (ISPEC) 2006, LNCS, vol. 3903, pp. 202–213. Springer (2006). doi:10.1007/11689522_19.
9. Chien H.Y.: ID-based key agreement with anonymity for ad hoc networks. In: Huo T.W., Sha E., Guo M., Yang L., Shao Z. (eds.) Proc. Embedded and Ubiquitous Computing (EUC) 2007, LNCS, vol. 4808, pp. 333–345. Springer (2007). doi:10.1007/978-3-540-77092-3_29.
10. Chow S.S.M., Choo K.K.R.: Strongly-secure identity-based key agreement and anonymous extension. In: Garay J., Lenstra A., Mambo M., Peralta R. (eds.) Proc. 10th International Conference on Information Security Conference (ISC) 2007, LNCS, vol. 4779, pp. 203–220. Springer (2007). doi:10.1007/978-3-540-75496-1_14.
11. Di Raimondo M., Gennaro R., Krawczyk H.: Deniable authentication and key exchange. In: Wright R., De Capitani de Vimercati S., Shmatikov V. (eds.) Proc. 13th ACM Conference on Computer and Communications Security (CCS), pp. 400–409. ACM (2006). doi:10.1145/1180405.1180454.

12. Dierks T., Allen C.: The TLS protocol version 1.0 (1999). http://www.ietf.org/rfc/rfc2246.txt. RFC 2246.
13. Dierks T., Rescorla E.: The Transport Layer Security (TLS) protocol version 1.2 (2008). http://www.ietf.org/rfc/rfc5246.txt. RFC 5246.
14. Diffie W., Hellman M.E.: New directions in cryptography. IEEE Trans. Inform. Theory **22**(6), 644–654 (1976).
15. Dingledine R., Mathewson N., Syverson P.: Tor: the second-generation onion router. In: Proc. 13th USENIX Security Symposium. The USENIX Association (2004). http://www.usenix.org/events/sec04/tech/dingledine.html.
16. Fiore D., Gennaro R., Smart N.P.: Constructing certificateless encryption and ID-based encryption from ID-based key agreement. In: Joye M., Miyaji A., Otsuka A. (eds.) Proc. Pairing-Based Cryptography (Pairing) 2010, LNCS, vol. 6487, pp. 167–186. Springer (2010). doi:10.1007/978-3-642-17455-1_11.
17. Goldberg I.: On the security of the Tor authentication protocol. In: Danezis G., Golle P. (eds.) Privacy Enhancing Technologies (PET) 2006, LNCS, vol. 4258, pp. 316–331. Springer (2006). doi:10.1007/11957454_18
18. Google: The Official Google Blog—search more securely with encrypted Google web search (2010). http://googleblog.blogspot.com/2010/05/search-more-securely-with-encrypted.html
19. Kate A., Zaverucha G.M., Goldberg I.: Pairing-based onion routing with improved forward secrecy. ACM Trans. Inform. Syst. Secur. **13**(4), 29 (2010). doi:10.1145/1880022.1880023.
20. Krawczyk H.: SIGMA: The 'SIGn-and-MAc' approach to authenticated Diffie-Hellman and its use in the IKE protocols. In: Boneh D. (ed.) Advances in Cryptology—Proc. CRYPTO 2003, LNCS, vol. 2729, pp. 400–425. Springer (2003). doi:10.1007/b11817. Full version available as http://www.ee.technion.ac.il/~hugo/sigma.ps.
21. Krawczyk H.: HMQV: A high-performance secure Diffie-Hellman protocol. In: Cramer R. (ed.) Advances in Cryptology—Proc. CRYPTO 2005, LNCS, vol. 3621, pp. 546–566. Springer (2005). doi:10.1007/11535218_33.
22. LaMacchia B., Lauter K., Mityagin A.: Stronger security of authenticated key exchange. In: Susilo W., Liu J.K., Mu Y. (eds.) First International Conference on Provable Security (ProvSec) 2007, LNCS, vol. 4784, pp. 1–16. Springer (2007). doi:10.1007/978-3-540-75670-5_1.
23. Law L., Menezes A.J., Qu M., Solinas J., Vanstone S.: An efficient protocol for authenticated key agreement. Des. Codes Cryptogr. **28**, 119–134 (2003). doi:10.1023/A:1022595222606. Previously appeared as http://www.cacr.math.uwaterloo.ca/techreports/1998/corr98-05.pdf.
24. Menezes A.J., Ustaoglu B.: Comparing the pre- and post-specified peer models for key agreement. Int. J. Appl. Cryptogr. **1**(3), 236–250 (2009). doi:10.1504/IJACT.2009.023472.
25. Menezes A., van Oorschot P.C., Vanstone S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton, FL, USA (1997).
26. Morrissey P., Smart N.P., Warinschi B.: A modular security analysis of the TLS handshake protocol. In: Pieprzyk J. (ed.) Advances in Cryptology—Proc. ASIACRYPT 2008, LNCS, vol. 5350, pp. 55–73 (2008). doi:10.1007/978-3-540-89255-7_5.
27. M'Raïhi D., Naccache D.: Batch exponentiations: a fast DLP-based signature generation strategy. In: Gong L., Stern J. (eds.) CCS 1996: Proceedings of the 3rd ACM Conference on Computer and Communications Security, pp. 58–61. ACM (1996). doi:10.1145/238168.238187.
28. NIST National Institute of Standards and Technology: Special Publication 800-56A, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography (2007). http://csrc.nist.gov/publications/PubsSPs.html.
29. NIST National Institute of Standards and Technology: Special Publication 800-56B, Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography (2009). http://csrc.nist.gov/publications/PubsSPs.html.
30. OpenSSL Project, The: OpenSSL v1.0.0d (2011). http://www.openssl.org/.
31. Øverlier L., Syverson P.: Improving efficiency and simplicity of Tor circuit establishment and hidden services. In: Privacy Enhancing Technologies, LNCS, vol. 4776, pp. 134–152. Springer (2007). doi:10.1007/978-3-540-75551-7_9.
32. Pfitzmann A., Hansen M.: A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management (2010). http://dud.inf.tu-dresden.de/Anon_Terminology.shtml. V0.34.
33. Rahman S.M.M., Inomata A., Okamoto T., Mambo M., Okamoto E.: Anonymous secure communication in wireless mobile ad-hoc networks. In: Stajano F., Kim H.J., Chae J.S., Kim S.D. (eds.) Proc. International Converence on Ubiquitous Convergence Technology (ICUCT) 2006, LNCS, vol. 4412, pp. 140–149. Springer (2007). doi:10.1007/978-3-540-71789-8_15.
34. Shoup V.: On formal models for secure key exchange (version 4) (1999). http://shoup.net/papers/skey.pdf.

35. Singel R.: Charter to snoop on broadband customers' web histories for ad networks (2008). http://www.wired.com/threatlevel/2008/05/charter-to-inse/.
36. Slashdot: ISPs inserting ads into your pages (2007). http://yro.slashdot.org/yro/07/06/23/1233212.shtml.
37. Tor Project: Homepage (2011). http://www.torproject.org/.