A Zero-Knowledge Identification and Key Agreement Protocol

D.R. Stinson* and J. Wu†
David R. Cheriton School of Computer Science, University of Waterloo
200 University Ave. W., Waterloo, Ontario, N2L 3G1, Canada
{dstinson, j32wu}@uwaterloo.ca

Abstract

In this paper, we propose a zero-knowledge authenticated key agreement protocol with key confirmation (AKC) in asymmetric setting. The protocol has several desirable security attributes like some classical AKCs such as STS [7] and MQV [13]. One highlight of our protocol is its zero-knowledge property, which enables succinct proofs of the claimed security attributes, while the overhead in communication and computation resulting from the special design to achieve zero-knowledge is insignificant.

Keywords: mutual identification, key agreement, zero-knowledge.

1 Introduction

An *authenticated key agreement protocol* (AKC) enables two parties, Alice (A) and Bob (B), to establish a shared secret key via unsecured communication channels. Each run of the protocol produces a unique session key between A and B. Both parties are assured that (if their intended peer acts honestly and follows the protocol[1]) no third party can possibly compute the session key (this property is called *key authentication*). Furthermore, if they are assured that their peer has possession of the session key (assuming their intended peer acts honestly and follows the protocol), then the protocol is said to provide *explicit key confirmation*. If they are assured that their peer can compute the session key (assuming their intended peer acts honestly and follows the protocol), then the protocol is said to provide *implicit key confirmation*.

An AKC can be viewed as a secure mutual identification scheme coupled with a secure distribution of a session key (see [3], [17, §11]). Therefore security analysis for an AKC can be divided into two steps: first, show that the AKC is a secure mutual identification scheme; and, second, show that the session key is secure.

A *mutual identification scheme* is secure if the only way an adversary can get an honest party to accept is by faithfully relaying the messages between the party and its intended peer. In another words, the scheme is secure if no honest party will accept after the adversary is active, i.e., if he drops, injects, switches, alters messages, or does anything other than faithfully relaying the messages. Depending on the communication capabilities granted to the adversary, the attacks can be divided into several types:

*research supported by NSERC discovery grant 203114-06

†research supported by an NSERC post-graduate scholarship

[1] A dishonest peer may simply give away a session key to a third party, so we do not specify any outcome about the secrecy of the session key if an intended peer acts dishonestly or does not follow the protocol.

1. *passive attacks*, in which the adversary can only observe the messages exchanged between legitimate parties before trying to impersonate one.

2. *concurrent attacks*, in which the adversary can carry out the protocol concurrently with legitimate parties *before* trying to impersonate one.

3. *active-intruder attacks*, in which the adversary can carry out the protocol with legitimate parties *while* trying to impersonate one.

Note that active-intruder attacks cover concurrent attacks, which cover passive attacks in turn. In some formal security models for mutual identification schemes, e.g., the Bellare-Rogaway model [3], Blake-Wilson-Menezes model [13, 4], and Canetti-Krawczyk model [5], an active-intruder attack is captured using an equivalent idea named *matching conversation*, or a somewhat more formal notion, *matching session IDs*.

The security of a session key is defined in term of indistinguishability, i.e., a session key is indistinguishable from a key randomly drawn from the key space to an adversary when the adversary observes all the messages between the two parties. Because the protocol is a secure mutual identification scheme, an active adversary will cause a key agreement session to be aborted and no session key will be established. So we only need to consider a passive adversary for session key security.

In the above part, the capabilities of an adversary are discussed based only on the communication model. The adversary may have some capabilities other than eavesdropping and manipulating messages, e.g., the adversary may somehow get some session keys, or public keys of the legitimate parties (in the asymmetric setting). In such cases, the possible damage should be limited to the maximum possible extent. Some other attacks should also be taken into consideration. A number of desirable attributes for AKCs have been identified in the literature (see, e.g., [13, 4]), including

1. *known session key secrecy*: if some session keys are revealed to an adversary, other unrevealed session keys are still secure.

2. *perfect forward secrecy*: if the long-term private keys of one or more entities are compromised, the secrecy of previous session keys is not affected.

3. *key compromise impersonation resilience*: when A's long-term private key is disclosed, this loss does not enable an adversary to impersonate other entities to A.

4. *unknown key share resilience*: A cannot be coerced into sharing a key with entity B without A's knowledge.

To achieve the above security features is not an easy task. The design of AKC protocols is extremely error prone. The literature is filled with protocols which have been found to contain certain security flaws. In [3], Bellare and Rogaway first proposed a formal security model for AKCs. Since then, there have been several extensions to the model, e.g., the Blake-Wilson-Menezes model [13, 4] for the asymmetric setting, and Bellare-Canetti-Krawczyk's modular model [1], which is a construction approach as well as a proof model. A formal proof with respect to such cryptographic definitions provides a strong assurance that a protocol is behaving as desired.

Extending the zero-knowledge unilateral identification scheme presented in [18], in this paper we propose an authenticated key agreement protocol with key confirmation in the asymmetric

Protocol 2.1: Protocol Setup

1. The *TA* chooses a large prime p such that $p-1$ is divisible by another large prime q, $\log_2 q \approx k$, and $\log_2 p \approx k'$, where k' is polynomial in k and k is a security parameter.
2. The *TA* chooses an element $g \in {\mathbb{Z}_p}^*$ having order q.
3. The *TA* publishes the triple (p, q, g).
4. The *TA* publishes a hash function $h : \{0,1\}^* \rightarrow \{0,1\}^k$.
5. Each potential prover A chooses a private key a from ${\mathbb{Z}_q}^*$ uniformly at random, computes $v = g^a \mod p$, and registers v as his or her public key.
6. The *TA* verifies that $v \neq 1$ and that A possesses the private key a. If both of these facts are verified, then the *TA* issues a certificate to A certifying that v is indeed A's public key.

setting. The protocol provides the security attributes listed above. One highlight of the protocol is its zero-knowledge property, which enables succinct proofs of these claimed security attributes. Meanwhile the overhead in communication and computation resulting from the special design to achieve zero-knowledge is insignificant. The security proof is based on a security definition different from but complying with the popular security definition in [3], which simplifies the proof for our protocol.

The remainder of the paper is organized as follows: section 2 describes the protocol; section 3 discusses its zero-knowledge property; section 4 proves the protocol is a secure mutual identification scheme; section 5 proves the protocol is a secure AKC; section 6 discusses its computation and communication performance; and section 7 concludes this paper.

2 Protocol Definition

2.1 Initial Setup

The initial setup for our scheme is described in Protocol 2.1. (Observe that the setup of the scheme is defined in terms of a security parameter k.) We assume the existence of a trusted authority, denoted by *TA*, who will issue certificates for all potential participants in the scheme. Furthermore, we assume that every potential prover proves to the *TA* that he or she knows the private key corresponding to the public key that is being registered.

2.2 Protocol Description

In a session of the scheme, two parties A and B try to convince each other of their identities. Each party "accepts" only if the other one responds to challenges in an appropriate way. Upon acceptance, each party computes a session key.

We are working in the *pre-specified peer model*. This means that, before a session of the protocol is executed, the two parties exchange certificates. There is no guarantee that the certificates are

Protocol 2.2: Protocol Description

1. B chooses $r_1 \in {\mathbb{Z}_q}^*$ uniformly at random, computes

$$y_1 = g^{r_1} \mod p \text{ and } h_1 = h({v_a}^{r_1} \mod p),$$

and sends (y_1, h_1) to A.

2. A verifies (y_1, h_1):
If $y_1 \notin \{2, \ldots, p-1\}$ or $1 \neq {y_1}^q \mod p$ or $h_1 \neq h({y_1}^a \mod p)$, then A rejects.
Otherwise A chooses $r_2 \in {\mathbb{Z}_q}^*$ uniformly at random, computes

$$y_2 = g^{r_2} \mod p \text{ and } h_2 = h({v_b}^{r_2} \mod p \parallel {y_1}^a \mod p),$$

and sends (y_2, h_2) to B.

3. B verifies (y_2, h_2) :
If $y_2 \notin \{2, \ldots, p-1\}$ or $1 \neq {y_2}^q \mod p$ or $h_2 \neq h({y_2}^b \mod p \parallel {v_a}^{r_1} \mod p)$, then B rejects.
Otherwise B accepts, computes

$$h_3 = h({y_2}^b \mod p) \text{ and } K = {y_2}^{r_1} \mod p,$$

sends h_3 to A, and keeps K as the established session key.

4. A verifies h_3. If $h_3 \neq h({v_b}^{r_2} \mod p)$, then A rejects. Otherwise A accepts and computes

$$K = {y_1}^{r_2} \mod p$$

as the session key.

exchanged securely. For example, an adversary may substitute one certificate with another one. However, before any party, say A, engages in a session, we assume that A has received and verified the certificate of a party, say B, who will be termed the *intended peer* of A. If the intended peer of A is B, it does not necessarily follow that the intended peer of B is A.

The steps in a session are described in Protocol 2.2. In each session a transcript is generated. The transcript consists of the messages exchanged between the two parties. For example, in a session in which both parties accept, the transcript is $(y_1, h_1, y_2, h_2, h_3)$. When one party rejects, we denote the messages afterwards as $\perp$. For example, if A rejects (y_1, h_1), then we denote the transcript as $(y_1, h_1, \perp)$. In the following parts we omit the operation "mod p" to simplify the expression.

Assuming both parties follow the protocol, the message flows can be depicted as follows:

$$
\begin{array}{ccc}
A & \xleftarrow{y_1=g^{r_1},\, h_1=h({v_a}^{r_1})} & B \\
A & \xrightarrow{y_2=g^{r_2},\, h_2=h({v_b}^{r_2} \| {y_1}^{a})} & B \\
A & \xleftarrow{h_3=h({y_2}^{b})} & B
\end{array}
$$

2.3 Completeness

It is straightforward to verify that Protocol 2.2 is complete. Suppose A and B are both honest. First B sends out $(y_1 = g^{r_1}, h_1 = h({v_a}^{r_1}))$. After receiving the challenge (y_1, h_1), A checks if ${y_1}^{q} = 1$ and $h({y_1}^{a}) = h_1$. Since

$$h({y_1}^{a}) = h(g^{ar_1}) = h({v_a}^{r_1}) = h_1,$$

A accepts and sends the response $(y_2 = g^{r_2}, h_2 = h({v_b}^{r_2} \parallel {y_1}^{a}))$ to B. B then checks if ${y_2}^{q} = 1$ and $h({y_2}^{b} \parallel {v_a}^{r_1}) = h_2$. Since

$$h({y_2}^{b} \parallel {v_a}^{r_1}) = h({v_b}^{r_2} \parallel {y_1}^{a}) = h_2,$$

B also accepts, computes his session key $K_b = {y_2}^{r_1}$, and sends $h({y_2}^{b}) = h_3$ to A. Since

$$h({v_b}^{r_2}) = h({y_2}^{b}) = h_3,$$

A will accept and compute her session key $K_a = {y_1}^{r_2} = {y_2}^{r_1} = K_b$.

3 Zero-Knowledge Properties

In a session of the protocol, each party acts as a prover to convince the other party of his or her identity, and also as a verifier to verify the identity of the other party as well. We show that, for an honest verifier, the protocol is perfect zero-knowledge, and for an arbitrary (including a dishonest) verifier, the protocol is statistical zero-knowledge assuming that the KEA assumption (see Definition 3.3) holds and the hash function $h()$ used in the protocol is a random oracle.

3.1 Perfect Zero-Knowledge for Honest Verifiers

First we consider A as a prover and B as a verifier. Suppose B is honest. The real transcripts between A and B can be simulated by a polynomial time simulator S without participating a real session, and the distributions of the real transcripts and the simulated transcripts are identical:

Theorem 3.1. *If A is considered as an honest prover and B is considered as an honest verifier, Protocol 2.2 is perfect zero-knowledge for B.*

Proof. The set $\mathcal{T}$ of real transcripts obtained by A and B consists of all transcripts T having the following form:

$$
\begin{aligned}
T &= (y_1, h_1, y_2, h_2, h_3) \\
&= (g^{r_1}, h({v_a}^{r_1}), g^{r_2}, h({v_b}^{r_2} \parallel {y_1}^{a}), h({y_2}^{b})) \\
&= (g^{r_1}, h(g^{ar_1}), g^{r_2}, h(g^{br_2} \parallel g^{ar_1}), h(g^{br_2}))
\end{aligned}
$$

where r_1 and r_2 are chosen uniformly from $\mathbb{Z}_q^*$ at random independently.

The set $\hat{\mathcal{T}}$ of simulated transcripts can be constructed by S as follows. S chooses r_1 and r_2 uniformly independently at random from $\mathbb{Z}_q^*$, and computes the simulated transcript $\hat{T}$, using g, v_a, v_b, and $h()$:

$$\begin{aligned}\hat{T} &= (g^{r_1}, h({v_a}^{r_1}), g^{r_2}, h({v_b}^{r_2} \parallel {v_a}^{r_1}), h({v_b}^{r_2})) \\ &= (g^{r_1}, h(g^{ar_1}), g^{r_2}, h(g^{br_2} \parallel g^{ar_1}), h(g^{br_2})).\end{aligned}$$

Since the randomly chosen r_1 and r_2 in T and $\hat{T}$ have identical probability distributions, the transcripts $\mathcal{T}$ and $\hat{\mathcal{T}}$ have identical probability distributions, and the protocol is perfect zero-knowledge for B. □

Similarly, when A is considered as an honest verifier in the protocol, S can simulate the real transcripts in the same way, therefore we have

Theorem 3.2. *If B is considered as an honest prover and A is considered as an honest verifier, Protocol 2.2 is perfect zero-knowledge for A.*

Proof. As in the proof of Theorem 3.1, the set $\mathcal{T}$ of real transcripts obtained by A and B consists of all transcripts T having the following form:

$$T = (g^{r_1}, h(g^{ar_1}), g^{r_2}, h(g^{br_2} \parallel g^{ar_1}), h(g^{br_2}))$$

where r_1 and r_2 are chosen uniformly from $\mathbb{Z}_q$ at random independently.

S chooses r_1 and r_2 uniformly at random from $\mathbb{Z}_q$, and computes the simulated transcript $\hat{T}$, using g, v_a, v_b, and $h()$:

$$\hat{T} = (g^{r_1}, h(g^{ar_1}), g^{r_2}, h(g^{br_2} \parallel g^{ar_1}), h(g^{br_2})).$$

The transcripts $\mathcal{T}$ and $\hat{\mathcal{T}}$ have identical probability distributions, and the protocol is perfect zero-knowledge for A. □

3.2 Statistical Zero-Knowledge for Arbitrary Verifiers

Now we consider the case in which a verifier might be dishonest. A dishonest verifier may generate messages not following the protocol. First let A be a prover and B be an arbitrary (including an dishonest) verifier. B may violate the protocol during a session, e.g., he may generate (y_1, h_1) without choosing r_1, or he might choose r_1 with a nonuniform probability distribution. We show that the protocol is statistical zero-knowledge for B if the KEA assumption holds and the hash function $h()$ is a random oracle.

KEA is a non-standard cryptographic assumption first proposed by Damgård in [6]. Its application can be found in various papers, e.g., [2, 11, 12, 18]. Although it is a strong assumption, it seems to hold in most groups where the computation of discrete logarithms is intractable. Informally, KEA says that, in a prime order group $\langle g \rangle$, given g^a with a unknown, the only way (except with negligible probability) to construct in polynomial time a pair of the form (d, d^a) is to first compute $d = g^r$ for some r, and then compute $d^a = (g^a)^r$. More formally, the assumption is stated in the next definition.

Definition 3.3. *The* Knowledge-of-Exponent Assumption *is as follows: Let g be a generator of a multiplicative (sub)group. For any polynomial-time algorithm $\boldsymbol{A}$ that takes as input g and g^a where a is an unknown randomly chosen value, and which produces as output a pair of the form $(d, d'), d \in \langle g \rangle$, there exists a polynomial-time "extractor" $\boldsymbol{A}'$, which takes the same input and outputs the pair (d, d') along with an exponent r, such that for sufficiently large k, it holds that*

$$\Pr[d' = d^a \text{ and } g^r \neq d] < \frac{1}{Q(k)}$$

for any polynomial $Q()$.

The KEA implies that we have access to the internal state of an algorithm. Therefore a simulation based on KEA is non-black-box in nature. In a non-black-box simulation, the simulator has access to any internal information of a verifier, e.g., its private key and its control flow. However, the simulator must be polynomial time, and it does not have access to the prover (neither its internal information nor as an oracle) in the simulation.

For a dishonest verifier, we need to consider auxiliary-input zero knowledge. That is, the verifier may collect auxiliary input, e.g., some messages from previous sessions, and use them in a new session. In this case, we assume that the simulator has access to the transcripts of these previous sessions as well as the internal states of the verifiers while they were involved in producing the auxiliary input.

Based on the KEA assumption, we have the following result:

Theorem 3.4. *When A is considered as an honest prover and B as an arbitrary verifier, the protocol is auxiliary-input statistical zero-knowledge for B if the KEA assumption holds and $h()$ is a random oracle.*

Proof. S simulates transcripts as follows: first S receives the challenge pair (y_1, h_1) from B, then he completes the transcript as follows:

1. If $y_1 \notin \{2, \ldots, p-1\}$, or $1 \neq {y_1}^q$, or h_1 is not generated by a verifier querying $h()$, then S outputs the transcript
$$\hat{T} = (y_1, h_1, \perp).$$

2. If h_1 is a reply of $h()$, say $h_1 = h(z)$, which means a verifier C generates the pair $(y_1, z) \leftarrow \mathbf{A}(g, v_a)$ first, then S computes r_1 with the corresponding extractor $(y_1, z, r_1) \leftarrow \mathbf{A}'(g, v_a)$.[2]

   (a) If $(y_1, z) = (g^{r_1}, {v_a}^{r_1})$, then S chooses $r_2 \in {\mathbb{Z}_q}^*$ uniformly at random, computes
   $$y_2 = g^{r_2}, h_2 = h({v_b}^{r_2} \parallel z).$$
   S sends (y_2, h_2) to B and receives h_3, then output the transcript
   $$\hat{T} = (y_1, h_1, y_2, h_2, h_3).$$

   (b) Otherwise (i.e., $(y_1, z) \neq (g^{r_1}, {v_a}^{r_1})$), S outputs the transcript
   $$\hat{T} = (y_1, h_1, \perp).$$

[2] C does not has to be the same as B, because B may get y_1, h_1 from the auxiliary input.

From the structure of the protocol and the simulation process, we see that a difference between the distributions of the real transcripts $\mathcal{T} = \{T\}$ and the simulated transcripts $\hat{\mathcal{T}} = \{\hat{T}\}$ only happens when the challenge pair (y_1, h_1) is valid, i.e., $y_1 \in \{2, \ldots, p-1\}$, $1 = {y_1}^q$, and $h_1 = h({y_1}^a)$. At this time, the real transcript T is always $(y_1, h_1, y_2, h_2, h_3)$, while the simulated transcript $\hat{T}$ may be $(y_1, h_1, y_2, h_2, h_3)$ or $(y_1, h_1, \perp)$. In the following three cases, the real and simulated transcripts are different:

1. h_1 is not the reply of a query on $h()$ but $h_1 = h({y_1}^r)$.

   In this case, C happens to choose $h_1 = h({y_1}^a)$. The statistical distance between the probability distributions of T and $\hat{T}$ caused by this reason is $1/2^k$.

2. h_1 is the reply of a query on $h(z)$, $h(z) = h({y_1}^a)$ but $z \neq {y_1}^a$.

   In this case, C finds a second pre-image of $h({y_1}^a)$. The statistical distance between the probability distributions of T and $\hat{T}$ caused by this reason is no more than $1/2^k$.

3. h_1 is the reply of a query on $h(z)$, $z = {y_1}^a$ but $g^{r_1} \neq z$.

   In this case, C first generates a pair (y_1, z) such that $y_1 \in \langle g \rangle$ and $z = {y_1}^a$, but the corresponding extractor fails to find r_1 such that $g^{r_1} = y_1$. The statistical distance between the probability distributions of T and $\hat{T}$ caused by this reason is less than $1/Q(k)$ due to KEA assumption.[3]

Therefore, the statistical distance d between the probability distributions of T and $\hat{T}$ is

$$d < 1/Q(k) + 1/2^{k-1}.$$

□

Next we consider B as a prover and A as an arbitrary (including a dishonest) verifier. In this case A may violate the protocol during a session. We have the following result:

Theorem 3.5. *When B is considered as an honest prover and A as an arbitrary verifier, the protocol is statistical zero-knowledge for A if KEA assumption holds and $h()$ is a random oracle.*

Proof. S simulates the transcripts as follows: choose $r_1 \in {\mathbb{Z}_q}^*$ and compute

$$y_1 = g^{r_1}, h_1 = h({v_a}^{r_1}),$$

sends (y_1, h_1) to A, receives (y_2, h_2) returned by A, and complete the transcript as follows:

1. If $y_2 \notin \{2, \ldots, p-1\}$, or $1 \neq {y_2}^q$, or h_2 is not generated by A querying $h()$, then output the transcript
$$\hat{T} = (y_1, h_1, y_2, h_2, \perp).$$

[3]Note $y_1 \in \langle g \rangle$ is necessary for KEA to hold. If $y_1 \notin \langle g \rangle$, the proof breaks down. Correspondingly, if in the protocol, $y_1 \in \langle g \rangle$ is not checked, then the protocol will suffer from a small sub-group attack. The same reason holds for checking $y_2 \in \langle g \rangle$.

2. If h_2 is generated by A querying $h()$, say $h_2 = h(z)$, which means A generates the pair (y_2, z) first. If ${y_1}^a$ is not the suffix of z, then S outputs the transcript

$$\hat{T} = (y_1, h_1, y_2, h_2, \perp).$$

Otherwise ($z = s \parallel {y_1}^a$ for certain s), A must have constructed the pair $(y_2, s) \leftarrow \mathbf{A}(g, v_b)$. Then S computes r_2 with the corresponding extractor $(y_2, s, r_2) \leftarrow \mathbf{A}'(g, v_b)$.

(a) If $(y_2, s) = ({g}^{r_2}, {v_b}^{r_2})$, then S computes $h_3 = h({v_b}^{r_2})$, and outputs the transcript

$$\hat{T} = (y_1, h_1, y_2, h_2, h_3).$$

(b) Otherwise S outputs the transcript

$$\hat{T} = (y_1, h_1, y_2, h_2, \perp).$$

From the structure of the protocol and the simulation process, we see that a difference between the distributions of the real transcripts $\mathcal{T} = \{T\}$ and the simulated transcripts $\hat{\mathcal{T}} = \{\hat{T}\}$ only happens when the pair (y_2, h_2) is valid, i.e., $y_2 \in \{2, \ldots, p-1\}$, $1 = {y_2}^q$, and $h_2 = h({y_2}^b \parallel {y_1}^a)$. At this time, the real transcript T is always $(y_1, h_1, y_2, h_2, h_3)$, while the simulated transcript $\hat{T}$ may be $(y_1, h_1, y_2, h_2, h_3)$ or $(y_1, h_1, y_2, h_2, \perp)$. Using a similar argument as in Theorem 3.4, we conclude that the statistical distance d between the probability distributions of T and $\hat{T}$ is

$$d < 1/Q(k) + 1/2^{k-1}.$$

□

3.3 Concurrent Zero-knowledge

The zero-knowledge property of our protocol is preserved under three types of composition of sessions of the protocol, namely, *sequential composition*, *parallel composition*, and *concurrent composition*. In sequential composition, the protocol is invoked many times, where each invocation follows the termination of a previous one. Every zero-knowledge protocol is zero-knowledge under sequential composition. Under parallel composition, many instances of the protocol are invoked at the same time and they proceed at the same pace. In general, zero-knowledge properties are not preserved under parallel composition. Concurrent composition generalizes both sequential and parallel composition. In this setting, polynomially many instances of the protocol are invoked at arbitrary times and they proceed at an arbitrary pace [9].

With the same simulation techniques used in the proofs of Theorem 3.1, 3.2, 3.5 and 3.4, we can show the following results:

Theorem 3.6. *When A is considered as an honest prover and B as an honest verifier, concurrent composition of Protocol 2.2 preserves perfect zero-knowledge for B if KEA assumption holds and $h()$ is a random oracle.*

Theorem 3.7. *When B is considered as an honest prover and A as an honest verifier, concurrent composition of Protocol 2.2 preserves perfect zero-knowledge for A if KEA assumption holds and $h()$ is a random oracle.*

Theorem 3.8. *When A is considered as an honest prover and B as an arbitrary verifier, concurrent composition of Protocol 2.2 preserves statistical zero-knowledge for the B if KEA assumption holds and $h()$ is a random oracle.*

Theorem 3.9. *When B is considered as an honest prover and A as an arbitrary verifier, concurrent composition of Protocol 2.2 preserves statistical zero-knowledge for A if KEA assumption holds and $h()$ is a random oracle.*

Here we only give a proof for Theorem 3.9. Proofs for Theorem 3.6, 3.7, and 3.8 can be obtained in a similar way.

Proof. Let T be the transcript of a concurrent composition of the protocol between A and B. Then T is a composition of m individual transcripts, where m is polynomial in k. Let each of these individual transcripts be $((y_1, h_1)^i, (y_2, h_2)^i, (y_3)^i)$, where the superscript $i \in [1, m]$ is a unique ID of the transcript. We assume transcript 1 starts first and transcript m finishes last. Then

$$T = ((y_1, h_1)^1, \cdots, (y_1, h_1)^i, \cdots, (y_2, h_2)^i, \cdots, (h_3)^i, \cdots, (h_3)^m).$$

The position of each message is known to both A and B.

S simulates the composite transcript T as follows: for the individual transcript i, constructs $((y_1, h_1)^i, (y_2, h_2)^i, (h_3)^i)$ the same way as in Theorem 3.5. and puts $(y_1, h_1)^i$, $(y_2, h_2)^i$ and $(h_3)^i$ in the simulated transcript $\hat{T}$. The positions of these messages correspond to where they are supposed to be in a real transcript.

From the proof of Theorem 3.5, we know that the statistical distance d^i between the probability distributions of each pair of individual transcripts T^i and $\hat{T}^i$ is

$$d^i < 1/Q(k) + 1/2^{k-1}.$$

T and $\hat{T}$ are composition of m such individual transcripts. Therefore, the statistical distance d between the probability distributions of T and $\hat{T}$ is

$$d < m/Q(k) + m/2^{k-1}.$$

□

4 Secure Mutual Identification

In this section we show that the protocol is a secure mutual identification scheme. The security definition and proof for the mutual identification scheme are essentially the same as those for the unilateral identification protocol in [18]. Also we will show that the security definition complies with the Bellare-Rogaway model in [3].

The security of the protocol is based on the intractability of the Computational Diffie-Hellman (CDH) Problem. The CDH Problem is to compute the value of g^{ab}, given g, g^a and g^b. The CDH Assumption is defined as follows.

Computational Diffie-Hellman Assumption.
Let g be a generator for a group of order q, and let $k = \log_2 q$. The CDH Assumption is that there is no polynomial-time algorithm (i.e., poly-time in k) that can compute g^{ab} for a non-negligible

fraction of all possible pairs $a, b \in \mathbb{Z}_q$, when it is given g, g^a, and g^b as input. That is, for any polynomial-time algorithm $\mathbf{A}$, and for any polynomial $Q()$, it holds that

$$\Pr[\mathbf{A}(g^a, g^b) = g^{ab}] < \frac{1}{Q(k)}$$

for all sufficiently large k, where a and b are random numbers uniformly distributed over ${\mathbb{Z}_q}^*$.

Next, we observe that the following result is obvious from the structure of Protocol 2.2.

Theorem 4.1. *An adversary can impersonate a prover with public key $v = g^a$ with non-negligible probability if and only if the adversary can compute g^{ab}, when it is given a random challenge g^b, for a non-negligible fraction of all values $b \in {\mathbb{Z}_q}^*$. That is, there exists a polynomial-time algorithm $\mathbf{A}_v$ with respect to public key $v = g^a$, and a polynomial $Q_1()$, such that for any large large k, it holds that*

$$\Pr[\mathbf{A}_v(g^b) = g^{ab}] > \frac{1}{Q_1(k)},$$

where b is randomly chosen from ${\mathbb{Z}_q}^*$ and $k = \log_2 q$.

Remark. In the above theorem, the prover can be either B who initiates a mutual identification session, or A who responses to the initiation.

Here is the notion of security that we will use.

Definition 4.2. *An identification protocol is* secure *if there is no polynomial-time adversary who can impersonate a non-negligible fraction of all possible provers with non-negligible probability. That is, for any polynomial $Q()$, and for all sufficiently large k, it holds that*

$$\Pr[\boldsymbol{A}_v \textit{ exists}] < \frac{1}{Q(k)}$$

where $v = g^a$, a is randomly chosen from ${\mathbb{Z}_q}^*$ and $\boldsymbol{A}_v$ is as defined in Theorem 4.1.

4.1 Concurrent Attack

Theorem 4.3. *Protocol 2.2 is secure against concurrent attacks in the random oracle model under the CDH and KEA Assumptions.*

Proof. Suppose that the protocol is not secure against concurrent attacks. Because the protocol is concurrent zero-knowledge, an adversary can impersonate a prover in a session without participating in any other sessions of the scheme. Next, since the adversary can impersonate a non-negligible fraction of all possible provers, there exists a polynomial $Q_2()$ such that, for sufficiently large k,

$$\Pr[\mathbf{A}_v \ \ exists] > \frac{1}{Q_2(k)}$$

for randomly chosen $a \in {\mathbb{Z}_q}^*$, where $v = g^a$, $k = \log_2 q$, and $\mathbf{A}_v$ satisfies

$$\Pr[\mathbf{A}_v(g^b) = g^{ab}] > \frac{1}{Q_1(k)},$$

as defined in Theorem 4.1. Therefore, given a pair g^a and g^b where a and b are randomly chosen from ${\mathbb{Z}_q}^*$, the probability that the adversary can compute g^{ab} is at least

$$\begin{aligned} \Pr[\mathbf{A}_v \textit{ exists and } \mathbf{A}_v(g^b) = g^{ab}] &= \Pr[\mathbf{A}_v \textit{ exists}] \cdot \Pr[\mathbf{A}_v(g^b) = g^{ab}] \\ &> \frac{1}{Q_1(k)Q_2(k)} \end{aligned}$$

for any large k. This contradicts the CDH Assumption. □

4.2 Active-intruder attack

One model to capture the adversary's ability to carry out active attacks is the Bellare-Rogaway model [3]. In this model, an adversary O interacts with multiple legitimate entities. O can generate, switch, alter, drop, and delay messages, or do anything he likes when he interacts with these entities. The security in this model is defined in terms of matching conversations as follows.

A secure mutual identification scheme is the one in which

1. if two honest parties have matching conversations, they will both accept, and
2. the probability that one honest party accepts while his intended peer is not engaged in a matching conversation is negligible.

This is equivalent to the secure definition for mutual authentication from [17, §11.1] and [19].

Definition 4.4. *A secure mutual identification scheme is the one in which*

1. *if both parties are honest, then they both accept, and*
2. *if the adversary has been active, then the probability that any honest party accepts after a flow in which the adversary is active is negligible.*

The two definitions are equivalent in that there will not be a matching conversation if and only if the adversary is active. Next we prove the protocol is a secure mutual identification protocol in terms of Bellare-Rogaway model. We need to show that if an honest party A accepts, her intended peer B is engaged in a matching conversation (with overwhelming probability).

Theorem 4.5. *Suppose A is an honest participant whose intended peer is B. Suppose that A engages in a session in which she*

1. *receives and accepts $m_1 = (y_1, h_1 = h({y_1}^a))$,*
2. *sends $m_2 = (y_2 = g^y, h_2 = h({v_b}^y \parallel {y_1}^a))$, and finally*
3. *receives and accepts $m_3 = h_3 = h({v_b}^y)$.*

Then B will have engaged in a session in which his intended peer is A, and in which he

1. *sent m_1,*
2. *received and accepted m_2, and finally*
3. *sent m_3.*

Proof. Let O be a polynomial time algorithm which runs (possibly multiple) instances of A and B as oracles. Each oracle only interacts with O, and executes the protocol correctly. O knows A's and B's public keys, but does not know their private keys. O can initiate an oracle to start a new session, or send a challenge to an oracle to start a new session. O can manage sessions concurrently.

Suppose an A oracle completes a session $\mathcal{S}$ with view (m_1, m_2, m_3), where m_1, m_2 and m_3 are as defined above.

After receiving A's reply m_2 in the session $\mathcal{S}$, O responded with $m_3 = h_3 = h({v_b}^y)$. O cannot compute h_3 unless it can compute the input ${v_b}^y = {y_2}^b = g^{yb}$. However, due to the CDH, it is infeasible to compute g^{yb} given g^b and g^y. Therefore, O must have obtained h_3 from a B oracle. Furthermore, from the structure of the protocol, it is clear that m_3 can only be returned by a B oracle as the third flow in a session $\mathcal{S}'$ in which the second flow was $(g^y, h(g^{by} \parallel s))$, for some $s \in {\mathbb{Z}_p}^*$. Because y is a fresh random value chosen by A, it holds with overwhelming probability that $s = {y_1}^a$ and the above-mentioned second flow of the session $\mathcal{S}'$ is indeed m_2.

Now, in the session $\mathcal{S}'$, the B oracle sent $m_3 = g^{by}$ in response to $m_2 = (g^y, h(g^{by} \parallel {y_1}^a))$. Suppose in $\mathcal{S}'$, B's intended peer is Z with a private key z and a public key V_z. Let B's first message in $\mathcal{S}'$ be $(g^{x'}, h({V_z}^{x'}))$, then ${V_z}^{x'} = {y_1}^{ax}$, which implies $zx' = ax$.

x' is a fresh random number chosen by B. z and a are private keys which were determined before the session $\mathcal{S}'$. z and x may be chosen by the adversary, but the adversary does not know x' or a. If $zx' = ax$ holds for random x', then there are two possible cases:

- $z = a$

  In this case, $x = x'$. B's intended peer is A, and B's first message is m_1.

- $z \neq a$

  In this case, only $z = 0, x = 0$ can make $zx' = ax$ hold for random x' and unknown $a \neq 0$. But this case is precluded by the protocol checking $y_1 \neq 1$ and $V_z \neq 1$.

Therefore, it holds with overwhelming probability that in session $\mathcal{S}'$, B's intended peer is A, and m_1 is the first flow in $\mathcal{S}'$.[4]

This completes the proof. □

Using similar methods, it is possible to prove a similar result concerning B's view of a session of the protocol.

Theorem 4.6. *Suppose B is an honest participant whose intended peer is A. Suppose that B engages in a session in which he*

1. *sends $m_1 = (y_1 = g^x, h_1 = h({v_a}^x))$,*
2. *receives and accepts $m_2 = (y_2, h_2 = h({y_2}^b \parallel {v_a}^x))$, and finally*
3. *sends $m_3 = h_3 = h({y_2}^b)$.*

Then A will have engaged in a session in which her intended peer was B, and in which she

1. *received and accepted m_1,*
2. *sent m_2, and finally*
3. *received and accepted m_3.*

[4] If $V_z \neq 1$ and $y_1 \neq 1$ are not verified in the protocol, the the proof breaks down. Correspondingly, the protocol will suffer from an impersonation attack. The same reason holds for checking $y_2 \neq 1$.

5 Key Agreement

After a successful mutual identification session using Protocol 2.2, A and B agree on a session key $K = g^{r_1 r_2}$. To be an authenticated key agreement scheme, the protocol must satisfy the following properties [17, §11]:

1. the protocol is a secure mutual identification scheme,
2. if the two parties engaged in the session are honest, then no information about the session key K can be computed by the adversary.

The attack model considered here includes passive/concurrent/active-intruder attacks. We have proved the first property in section 4. For the second property, since an active-intruder attack will cause a session to be aborted and no session key will be established, we only need to consider passive and concurrent attacks. Next we formally define and prove the second property.

5.1 Secure Authenticated Key Agreement with Key Confirmation

We define the security of a key agreement under passive and concurrent attacks as follows:

Definition 5.1. *A key agreement protocol is secure against passive and concurrent attacks if any polynomial time adversary cannot distinguish the established session key of a target session involving two honest participants from a random key drawn from the key space.*

Based on KEA assumption we have proved that the protocol is concurrent zero-knowledge. If the adversary can succeed using a concurrent attack, then he can succeed using a passive attack. Next we show that a passive attacker cannot distinguish a session key from a random key drawn from the key space. We prove this property based on the Decision Deffie-Hellman (DDH) assumption.

For the group $\langle g \rangle$ used in our protocol, DDH assumption states that given random $r_1, r_2, r \in \mathbb{Z}_q$, it is intractable to distinguish $(g^{r_1}, g^{r_2}, g^{r_1 r_2})$ from $(g^{r_1}, g^{r_2}, g^{r})$, i.e., for any polynomial time algorithm $\mathbf{A}$ and any polynomial Q, if k is sufficiently large, then

$$|\Pr[\mathbf{A}(g^{r_1}, g^{r_2}, g^{r_1 r_2}) = 1] - \Pr[\mathbf{A}(g^{r_1}, g^{r_2}, g^{r}) = 1]| < 1/Q(k).$$

Based on DDH, by reduction, it is easy to prove that for random $r_1, r_2, r, a, b \in \mathbb{Z}_q$,

$$(g^a, g^b, g^{r_1}, g^{r_2}, h(g^{a r_1}), h(g^{b r_2}), h(g^{b r_2} \parallel g^{a r_1}), g^{r_1 r_2})$$

and

$$(g^a, g^b, g^{r_1}, g^{r_2}, h(g^{a r_1}), h(g^{b r_2}), h(g^{b r_2} \parallel g^{a r_1}), g^{r})$$

are computationally indistinguishable, i.e, $K = g^{r_1 r_2}$ is indistinguishable from a random key.

Since the protocol is a secure mutual identification scheme, it provides implicit key confirmation. Summarizing the above discussion, we have the following result:

Theorem 5.2. *The protocol is an authenticated key agreement protocol secure against passive, concurrent, and active-intruder attacks, and it provides implicit key confirmation to both parties, assuming the KEA assumption and DDH assumption hold, and $h()$ is a random oracle.*

5.2 Known Session Key Attacks

In a known session key attack, the adversary O is allowed to acquire the session keys of the sessions he observes and/or participates in. The objective of O is to determine the key of a target session from which he cannot acquire the key. The target session may happen before, after, or in parallel with other sessions. One restriction on O is that he can only request the key of a session from a party that accepts in the session.

Theorem 5.3. *The protocol is secure against known session key attacks, given that the DDH assumption and the KEA assumption hold, and $h()$ is a random oracle.*

Proof. Let S_t be the target session, and T_t be the transcript of that session. In this proof, denote $[1, m] = \{1, \ldots, m\}$. Let $\{(T_i, K_i)|i \in [1, m]\}$ be the transcript and session key pairs that O accumulated by observing and/or participating in some sessions identified by integers $i \in [1, m]$, where m is polynomial in the security parameter k. If O can succeed in the known session key attack, then there must be a polynomial time algorithm $\mathbf{A}$ such that $\mathbf{A}(\{(T_i, K_i)|i \in [1, m]\}, T_t)$ can compute some partial information about K_t.

First observe that with r_1, r_2 and the public keys of the session participants, O can compute a simulated $(\hat{T}, \hat{K})$ pair without observing or participating in a real session:

$$\hat{T} = (g^{r_1}, h({v_a}^{r_1}), g^{r_2}, h({v_b}^{r_2} \parallel {v_a}^{r_1}), h({v_b}^{r_2})),\ \hat{K} = g^{r_1 r_2}. \tag{1}$$

Next we show that the probability distributions of $\{(T_i, K_i)|i \in [1, m]\}$ and a simulated $\{(\hat{T}_i, \hat{K}_i)|i \in [1, m]\}$ can be statistically indistinguishable.

O constructs a $(\hat{T}_i, \hat{K}_i)$ pair according to how the corresponding (T_i, K_i) pair is generated:

1. (T_i, K_i) is from an observed session between A and B.

   O simulates the transcript by choosing random r_1 and r_2, and computing $(\hat{T}_i, \hat{K}_i)$ using (1). Since r_1 and r_2 in (T_i, K_i) and $(\hat{T}_i, \hat{K}_i)$ have identical probability distributions, $(\hat{T}_i, \hat{K}_i)$ and (T_i, K_i) have identical probability distributions too.

2. (T_i, K_i) is from a session that O initiates and A accepts.

   Since the other party A accepts (y_1, h_1), with overwhelming probability O can obtain r_1 such that $y_1 = g^{r_1}$ due to KEA assumption. By choosing random r_2, O can complete the transaction $(\hat{T}_i, \hat{K}_i)$ using (1) without interacting with A. The probability distributions of (T_i, K_i) and $(\hat{T}_i, \hat{K}_i)$ are statistically indistinguishable. The detailed analysis is similar to that of Theorem 3.4.

3. (T_i, K_i) is from a session in which O participates, and which is initiated and accepted by B.

   Since B accepts (y_2, h_2), with overwhelming probability O can obtain r_2 such that $y_2 = g^{r_2}$. By choosing random r_1, O can construct $(\hat{T}_i, \hat{K}_i)$ using (1) without interacting with B. The probability distributions of (T_i, K_i) and $(\hat{T}_i, \hat{K}_i)$ are statistically indistinguishable. The detailed analysis is similar to that of Theorem 3.5.

Therefore O can build simulated $\{(\hat{T}_i, \hat{K}_i)|i \in [1, m]\}$ which are statistically indistinguishable from $\{(T_i, K_i)|i \in [1, m]\}$.

When $\mathbf{A}$ is a deterministic algorithm, it is obvious that the probability distributions of outputs of $\mathbf{A}(\{(\hat{T}_i, \hat{K}_i)|i \in [1, m]\}, T_t)$ and $\mathbf{A}(\{(T_i, K_i)|i \in [1, m]\}, T_t)$ are statistically indistinguishable.

When $\mathbf{A}$ is a randomized algorithm, it is equivalent to a deterministic algorithm $\mathbf{A_d}$ which has an additional random input r, where r has the same probability distribution as the random numbers within $\mathbf{A}$. When $(\{(\hat{T}_i, \hat{K}_i)|i \in [1,m]\}, T_t)$ and $(\{(T_i, K_i)|i \in [1,m]\}, T_t)$ are statistically indistinguishable, $(\{(\hat{T}_i, \hat{K}_i)|i \in [1,m]\}, T_t, r)$ and $(\{(T_i, K_i)|i \in [1,m]\}, T_t, r)$ are also statistically indistinguishable, and the outputs of $\mathbf{A_d}(\{(\hat{T}_i, \hat{K}_i)|i \in [1,m]\}, T_t, r)$ and $\mathbf{A_d}(\{(T_i, K_i)|i \in [1,m]\}, T_t, r)$ are statistically indistinguishable. So the probability distributions of the outputs of $\mathbf{A}(\{(\hat{T}_i, \hat{K}_i)|i \in [1,m]\}, T_t)$ and $\mathbf{A}(\{(T_i, K_i)|i \in [1,m]\}, T_t)$ are also statistically indistinguishable.

Therefore, if O can succeed in known session key attacks, then with negligible difference in probability, he can also succeed in a completely passive attack in which no session other than the target session takes place. This contradicts Theorem 5.2. □

5.3 Other Attributes of the Protocol

The protocol also provides the following desired attributes for key agreement protocols:

- *perfect forward secrecy.*

  Since a session key $K = g^{r_1 r_2}$ is independent of the private keys a and b, disclosing a or b does not help to gain any information about previous K.

- *key compromise impersonation resilience and unknown key share resilience.*

  Since the protocol is a secure mutual identification scheme, it is secure against key compromise impersonation and unknown key share attacks.

6 Performance

Comparing our protocol with MQV (AKC version) [13] and STS [7], we can see they have a similar structure in terms of exchanged messages, except that in our protocol, the first message is a challenge pair consisting of $y_1 = g^{r_1}$ and $h_1 = h({v_a}^{r_1})$, while in MQV and STS the first message only contains g^{r_1}. Computing and sending h_1 incurs additional communication and computation overhead. Suppose all these protocols are implemented on a subgroup of ${\mathbb{Z}_p}^*$. If we use SHA-1 as the hash function $h()$, then the additional communication overhead incurred by y_1 is 160 bits. Usually p needs to be at least 1024 bits. Therefore the increase in communication is insignificant. In view of computation, in all these protocols, in each session, each participant needs to do several exponentiation operations, which are the main computational overhead. Since computing or verifying h_1 incurs only one exponentiation and one hash operation for each party, the increase in the computation is also insignificant.

Our protocol can also be implemented in the setting of an elliptic curve E of prime order q, where $q \approx 2^{160}$. In this setting, the verifications that ${y_1}^q \bmod p \neq 1$ and ${y_2}^q \bmod p \neq 1$ are unnecessary; it would suffice to verify that y_1 and y_2 are points on E and that y_1 and y_2 are not the points at infinity. In the elliptic curve setting, the message length would be roughly 800 bits, and each party is required to perform only four "exponentiations" (i.e., scalar multiples of a point on the elliptic curve E).

7 Conclusion

In this paper, we proposed a zero-knowledge authenticated key agreement protocol with implicit key confirmation (AKC) in the asymmetric setting. The protocol has several desirable security attributes. One highlight of the protocol is its zero-knowledge property, which enables succinct proofs of the claimed security attributes, while the overhead in communication and computation resulting from the special design to achieve zero-knowledge is insignificant.

Acknowledgement

The authors thank Yunlei Zhao for pointing out some flaws in a previous version of our protocol in their technical report [20]. We have made some corrections to our protocol based on our correspondence with Zhao.

References

[1] M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. *Proceedings of STOC 98*, 419–428, ACM, 1998.

[2] M. Bellare and A. Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. Proceedings of Crypto 2004. *Lecture Notes in Computer Science* **3152**, 273–289, Springer-Verlag, 2004.

[3] M. Bellare and P. Rogaway. Entity authentication and key distribution. Proceedings Crypto 93. *Lecture Notes in Computer Science* **773**, 232–249, Springer-Verlag, 1994.

[4] S. Blake-Wilson, D. Johnson, and A.J. Menezes. Key agreement protocols and their security analysis. Proceedings of the sixth IMA International Conference on Cryptography and Coding. *Lecture Notes in Computer Science* **1355**, 30–45, Springer-Verlag, 1997.

[5] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. Proceedings of Eurocrypt 2001. *Lecture Notes in Computer Science* **2045**, 453–474, Springer-Verlag, 2001.

[6] I.B. Damgård. Towards practical public key systems secure against chosen ciphertext attacks, Proceedings of Crypto 91. *Lecture Notes in Computer Science* **576**, 445–456, Springer-Verlag, 1992.

[7] W. Diffie, P.C. van Oorschot, and M.J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes, and Cryptography*, **vol.2**, 107–125, 1992.

[8] U. Feige, A. Fiat, and A. Shamir. Zero knowledge proofs of identity. *Journal of Cryptology* **vol. 1**, 77–94, 1988.

[9] O. Goldreich. Zero-knowledge twenty years after its invention. Technical report, Department of Computer Science, Weizmann Institute of Science, 2002.

[10] L. Guillou and J. Quisquater. A "paradoxical" identity-based signature scheme resulting from zero-knowledge. Proceedings of Crypto 88. *Lecture Notes in Computer Science* **403**, 216–231, Springer-Verlag, 1990.

[11] S. Hada and T. Tanaka. On the existence of 3-round zero-knowledge protocols. Proceeding of Crypto 98. *Lecture Notes in Computer Science* **1462**, 408–423, 1998.

[12] H. Krawczyk. HMQV: A high-performance secure Diffe-Hellman protocol. Proceedings of Crypto 05. *Lecture Notes in Computer Science* **3621**, 546–466, 2005.

[13] L. Law, A.J. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated Key agreement. *Designs, Codes and Cryptography*, **vol. 28**, Issue 2, 119-134, 2003.

[14] T. Okamoto. Provably secure and practical identitication schemes and corresponding signature schemes. Proceeding of Crypto 92. *Lecture Notes in Computer Science* **740**, 31–53, Springer-Verlag, 1993.

[15] M.D. Raimondo, R. Gennaro, and H. Krawczyk, Deniable authentication and key exchange. Proceedings of the 13th ACM conference on Computer and communications security. 400–409, ACM Press, 2006.

[16] C.P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, **vol. 4**, 161–174, 1991.

[17] D.R. Stinson. *Cryptography: Theory and Practice, Third Edition*, Chapman & Hall/CRC, Boca Raton, 2006.

[18] D.R. Stinson and J. Wu. An efficient and secure two-flow zero-knowledge identification protocol, to appear in *Journal of Mathematical Cryptology*.

[19] H.B. Wang, Desired features and design methodologies of secure authenticated key exchange protoco in the public-key infrastructure setting. Master's Thesis, University of Waterloo, 2005.

[20] A.C.C. Yao, F.F. Yao, Y.L. Zhao, and B. Zhu, Deniable Internet Key-Exchange, http://eprint.iacr.org/2007/191.