An Efficient and Secure Two-flow Zero-Knowledge Identification Protocol

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

October 5, 2006

Abstract

In this paper, we propose a new zero-knowledge identification protocol. While the protocol consists of only two message flows, it does not rely on any underlying signature or encryption scheme. Its zero-knowledge property is preserved under concurrent composition and reset settings. It is secure under the strongest attack model which incorporates concurrent attacks, active-intruder attacks and reset attacks. Meanwhile its performance in computation and communication is close to that of the most efficient identification protocols not based on signature or encryption systems, most of which are insecure in this strong attack model.

1 Introduction

An identification protocol or scheme is an interactive protocol between a *prover*, "Alice" or "A", and a *verifier*, "Bob" or "B", in which the prover tries to identify herself to the verifier by demonstrating knowledge of a certain key associated with the prover. In the symmetric (secret key) setting, the key is shared between prover and the verifier, whereas in the asymmetric (public key) setting, the key is the private key of the prover. In this paper we are interested in the public key setting.

1.1 Challenge-response vs Customized Identification Techniques

Historically, there have been two distinct approaches to designing identification protocols. One approach is to start with an underlying signature scheme or public key encryption system. With this approach, in the public key setting, the prover demonstrates knowledge of her private key in one of two ways (see [20, §10.3.3]):

1. the prover decrypts a challenge from the verifier encrypted under her public key, or
2. the prover digitally signs a challenge from the verifier.

*research supported by NSERC discovery grant 203114-06
†research supported by an NSERC post-graduate scholarship

A protocol following this approach consists two message flows, consisting of the *challenge* sent from the verifier to the prover and the *response* sent from the prover to the verifier. The security of such a protocol relies on the underlying encryption or signature scheme.

The other approach is to design a protocol "from scratch" based on a specific computational problem such as the discrete logarithm problem. A large number of identification schemes following this approach make use of the *zero-knowledge proof of knowledge* technique, e.g., the Fiat-Shamir (FS) Scheme [10], the Schnorr Scheme [23], the Feige-Fiat-Shamir (FFS) Scheme [9], the Guillou-Quisquater (GQ) Scheme [15] and the Okamoto Scheme [21]. In these schemes, the prover proves her identity by means of a proof of knowledge, where the proof is zero-knowledge so as not to leak information about the private key. The security of the protocols relies on the presumed intractability of a specific computational problem. These schemes usually consists of three message flows: a *commitment* sent from the prover to the verifier, a *challenge* sent from the verifier to the prover, and a *response* sent from the prover to the verifier.

1.2 Attacks Models for Identification Protocols

Like any cryptographic protocol, an identification protocol needs to be secure against various kinds of adversaries, including legitimate but malicious users. The ultimate goal of an adversary is to impersonate the prover. Different types of attacks may be carried out depending on what capability is granted to the adversary. The weakest attack is eavesdropping, by which the adversary can only observe the communications between the prover and the verifier. But in many other scenarios, the adversary can do much more than that. The following attack models have been identified in the literature, capturing the capability of the adversary in different settings.

Sequential Attacks

In this model, the adversary can act as a verifier and interact with the prover in various session before the adversary tries to impersonate the prover. The sessions between the adversary and the prover are carried out sequentially. This is the standard attack model for the identification protocols in the smartcard communication model, and most existing zero-knowledge type identification protocols have been analyzed in this setting. In the literature, this kind of attack model is sometimes termed an "active attack" because the adversary can actively execute the protocol (as a verifier) in order to attempt to collect information beyond what he could obtain by passive eavesdropping. As we will see later, many other attacks came into consideration in which the adversary is even more "active", so here we call this attack a "sequential attack".

In the sequential attack model, the FFS and Okamoto schemes are secure assuming the intractability of the square-root problem and the discrete logarithm problem (DLP) respectively. There has been no proof of security for the Schnorr scheme and the GQ schemes under standard assumptions. However, based on the "one-more inversion" of DLP and RSA assumptions respectively, the schemes are proven secure in [3].

Concurrent Attacks

In sequential attack, the adversary can only execute multiple sessions sequentially. This is a natural restriction to the adversary in the smartcard communication model. But in the internet communication model, the adversary may have the ability to carry out multiple sessions with the prover, not

only sequentially, but also in parallel, i.e., concurrently. Previous research on zero-knowledge protocols has shown that concurrent execution of zero-knowledge protocols does not always preserve the zero-knowledge property, resulting in leakage of the prover's secret key. One such example is given in [14]. So it is natural to study the security of zero-knowledge based identification schemes under concurrent attacks.

The FFS, Okamoto, Schnorr and GQ schemes have been proven secure in this attack model under certain assumptions; see [3].

Active-intruder Attacks

In a concurrent attack, the adversary is only allowed to interact with the prover before he tries to impersonate the prover. But in the internet communication model, the adversary may be able to interact with the prover at the same time that he is impersonating the prover.

In such a setting, it is not straightforward to define what it means for an attack to be successful. For example, the adversary can faithfully relay all the messages between the prover and verifier, and the verifier will eventually accept. Yet this can hardly be regarded as a "real" attack. There are three approaches to handling this definitional issue. The first is based on the idea of *matching conversations* [8, 6], and a somewhat more formal notion is that of *matching conversation ids* [5]. The third approach is due to Stinson [24, §11.1] and uses the idea of an *active adversary*. Informally, an active adversary is an adversary who alters, injects, drops, and/or diverts at least one message between the prover and the verifier. (If the adversary simply relays messages between the prover and verifier, then the adversary is not active.) Based on this notion, we define a successful *active-intruder attack* as follows:

Definition 1.1. *In an active-intruder attack, the adversary is successful if the (honest) verifier accepts in a session after the adversary becomes active in the same session.*

We observe that the notions of matching conversations and an inactive adversary are equivalent, in that matching conversations take place if and only if there is no active adversary. Either notion (matching conversation or active adversary) includes concurrent attacks, as well. The active-intruder attack model is usually used in the context of identification schemes constructed from underlying encryption, signature or MAC schemes. Active-intruder attacks are also incorporated into security models for mutual identification protocols discussed in [1, 6]. The security model "concurrent reset 2" (CR2) for unilateral identification in [2] includes this attack.

In Appendix A we give examples of active-intruder attacks on several zero-knowledge based identification schemes. This shows that FFS, Schnorr, GQ and Okamoto schemes are not secure under this attack model.

Reset Attacks

In [2], Bellare *et al* proposed reset attacks on identification protocols, which originated in the context of zero-knowledge proofs. A reset attack allows the adversary to reset the prover's internal state. Such attacks are realistic when the prover is in the possession or control of the adversary, e.g., when the prover is a smartcard. For example, the adversary may not be able to read a secret key from the secure hardware of the smartcard, but it may be easy to reset the card by disconnecting the card's battery.

The following example shows how the adversary can extract the secret key in the Schnorr Scheme using a reset attack. The adversary resets the card, and starts an identification session. The card outputs a commitment x where $x = g^r$ and r is a random number generated by the card. The adversary then inputs a random number e_1 to the card, and the card will output $y_1 = ae_1 + r$ where a is the secret key of the card. Now the adversary can reset the card and start the second session. Since it is reset, the card will output the same $x = g^r$ as in the first session. The adversary then inputs the card with some value $e_2 \neq e_1$ and receives $y_2 = ae_2 + r$. Then the adversary can compute $a = (y_1 - y_2)/(e_1 - e_2)$. It has been observed that all proof-of-knowledge based identification protocols can be broken in a similar way [2].

The combination of active-intruder and reset attacks is the strongest attack model that has been considered for identification schemes to date. In [2], this attack model is named CR2. Correspondingly, there is an attack model termed CR1, which combines concurrent and reset attacks.

1.3 Generalized Paradigms For Identification

Bellare *et al* [2] proposed four paradigms for identification protocols, and analyzed their security against CR1 attacks as follows:

Signature based protocols

This refers to a signature based challenge-response identification protocol. The verifier sends a random challenge to the prover, and then the prover signs the challenge and sends the signature back to the verifier. The verifier then verifies the signature using the prover's public key. When the underlying signature scheme is secure, deterministic and stateless, the identification scheme is secure in the CR1 model.

Encryption based protocols

This refers to a public-key encryption based challenge-response identification protocol. The verifier encrypts a random challenge using the prover's public key and sends the ciphertext to the prover. The prover decrypts the ciphertext and sends the plaintext back to the verifier. When the underlying encryption scheme is secure against chosen-ciphertext attacks, the identification is secure in the CR1 model.

Identification based protocols

This refers to a general transformation of the Fiat-Shamir type proof-of-knowledge based identification protocols. The Fiat-Shamir type protocols are secure against concurrent attacks but are subject to reset attacks. The transformation utilizes resettable zero-knowledge techniques and turns three-flow Fiat-Shamir type protocols into four-flow protocols that are secure against both concurrent and reset attacks.

Resettable zero-knowledge based protocols

These protocols are based on resettable zero-knowledge arguments. Both the prover and the verifier are required to have public keys, and the protocol has a constant number of rounds (flows). The protocol is secure in the CR1 model.

In the CR2 setting, Bellare *et al* [2] proposed three paradigms:

1. a signature based protocol
2. an encryption based protocol.
3. an identification based protocol.

The three types of protocols are modifications of the corresponding ones in the CR1 setting. The modifications increase the message lengths. For encryption based protocols, the modification also increases the number of flows.

2-flow Zero-Knowledge Argument Systems

Another line of research work closely related to ours concerns zero-knowledge argument systems in the random oracle model. In [22], Pass shows that a 2-flow straight-line witness extractable deniable zero-knowledge argument system exists for any $\mathcal{NP}$-language in the random oracle model. However, the zero-knowledge argument system is very inefficient in communication complexity. In [11], Fischlin proposed a communication efficient straight-line witness extractable zero-knowledge proof which can improve the communication efficiency of Pass's 2-flow zero-knowledge argument, yet the improvement in communication is at the cost of large computational complexity. Considering the communication and computation complexity, it is not practical to use these zero-knowledge argument systems as identification schemes.

In [12], Furukawa *et al.* proposed a "compiler" that transforms a 3-flow honest verifier zero-knowledge protocol with special soundness property (a Σ-protocol) to a 2-flow deniable zero-knowledge argument. The communication and computation in one session of the protocol is only two to three times as much as in the underlying Σ-protocol (such as the Schnorr identification scheme). The efficiency of the constructed 2-flow zero-knowledge protocol makes it possible to be used as an identification protocol in practice.

1.4 Our Contributions

In this paper, we propose a 2-flow identification protocol, secure against CR2 attacks in the random oracle model, based on specific cryptographic assumptions (namely, the knowledge-of-exponent assumption and the computational Diffie-Hellman assumption). It is interesting to note that our protocol has the classical challenge-response structure, but it is not derived from the signature or encryption based design approach described above. It does not belong to any of the four paradigms for identification proposed in [2], either. While it is secure in the strongest security model considered to date, it is also very efficient. The total message length and computation required in one session are close to that of the Schnorr identification scheme, which is amongst the most efficient identification schemes used in practice.

Our work focuses on designing a specific identification scheme, instead of providing a general mechanism to transform any Σ-protocol to 2-move identification protocols, as is done in [22, 11, 12]. This results in a more efficient identification scheme.

Protocol 2.1: Identification Scheme Setup
Input: Security parameters k and k', which are positive integers.

1. The TA chooses a large prime p such that $p-1$ is divisible by another large prime q, where $\log_2 p \approx k'$, $\log_2 q \approx k$ and $q^2 \nmid (p-1)$.
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 $h : {\mathbb{Z}_p}^* \times {\mathbb{Z}_p}^* \to \{0,1\}^k$.
5. Each potential prover A chooses a private key a uniformly from $\mathbb{Z}_q$ at random, computes $v = g^a \mod p$, and registers v as his or her public key.
6. The TA issues a certificate to A certifying that v is indeed A's public key.

1.5 Organization of the Paper

The remainder of the paper is organized as follows: Section 2 describes the protocol; Section 3 gives the security analysis; Section 4 analyzes its computation and communication performance; and Section 5 concludes the paper.

2 Our New Identification Scheme

2.1 Initial Setup

The initial set-up for our scheme is described in Protocol 2.1. We assume the existence of a trusted authority, denoted by TA, who will issue certificates for all potential participants in the scheme. Observe that the setup of the scheme is defined in terms of security parameters k' and k. We would probably take $k' = 1024$ and $k = 160$ in practice.

2.2 Protocol Description

In a session of the scheme, the prover A tries to convince the verifier B of her identity. B "accepts" only if A responds to B's challenge in an appropriate way. The steps in a session of our scheme are summarized in Protocol 2.2. Observe that B's challenge must have a certain form; thus A "verifies" the challenge before she responds. After receiving A's response, B verifies it.

In the following, we omit the operation "mod p" to simplify the notation. Assuming that B's challenge is well-formed, the message flows can be depicted as follows:

$$\begin{array}{ccc} A & \xleftarrow{y_1 = g^r, h_1 = h(v^{rc} || y_1)} & B \\ A & \xrightarrow{y_2 = {y_1}^{ac}} & B \end{array}$$

Remark 2.1. We note that the structure of our protocol is similar to that of the "identification based on public-key decryption and witness" discussed in [20, §10.3.3]. As far as we know, the

Protocol 2.2: A 2-flow identification scheme

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

$$y_1 = g^r \bmod p \text{ and } h_1 = h((v^{rc} \bmod p)||y_1)$$

where $c = (p-1)/q$. Then B sends (y_1, h_1) to A.

2. After receiving (y_1, h_1), A rejects and stops if $y_1 = 0$, otherwise A computes

$$y_2 = {y_1}^{ac} \mod p.$$

If $h_1 = h(y_2||y_1)$ then A sends y_2 to B.

Otherwise A rejects and stops.

3. After receiving y_2, B verifies y_2. If $y_2 = v^{rc} \bmod p$, then B accepts; otherwise, B rejects.

first paper to suggest the general technique we are using in this paper is Damgård [7]. However, Damgård's paper just gives a brief description of the basic idea. For our protocol, we are going to give a detailed construction and thorough analysis, and we believe that our results provide the first proofs for schemes of this type.

2.3 Completeness

It is straightforward to prove that Protocol 2.2 is complete. Suppose A and B are both honest. After receiving the challenge (y_1, h_1), A checks to see if $h({y_1}^{ac}||y_1) = h_1$. Since

$$h({y_1}^{ac}||y_1) = h(g^{arc}||y_1) = h(v^{rc}||y_1) = h_1,$$

A accepts and sends the response $y_2 = {y_1}^{ac}$ to B. B then checks to see if $y_2 = v^{rc}$. Since

$$y_2 = {y_1}^{ac} = g^{arc} = v^{rc},$$

B also accepts.

2.4 Small Subgroup Attacks

First, we observe that $y^c \in \langle g \rangle$ for any $y \in {\mathbb{Z}_p}^*$. Furthermore, in the initial setup of the protocol, it is stipulated that $q^2 \nmid (p-1)$. Hence, $\gcd(c, p-1) = 1$ where $c = (p-1)/q$, which ensures that the mapping $y \mapsto y^c$ is a c-to-1 mapping from ${\mathbb{Z}_p}^*$ to $\langle g \rangle$. These properties allow the protocol to withstand a *small subgroup attack* (see [19]).

To illustrate, consider a modified scheme where h_1 is defined as $h_1 = h(v^r||y_1)$ and the prover checks if $h({y_1}^a||y_1) = h_1$. This modified protocol would be subject to a small subgroup attack, as follows. The adversary sends the challenge

$$y_1 = -g^r, h_1 = h(v^r||y_1)$$

to A. If he receives a reply from A, then he knows that A's secret key, namely a, is even; otherwise, a is odd. This yields one bit of information about the value of the secret key a.

This attack can be avoided by checking if $y_1 \in \langle g \rangle$ by A, but in doing so, the verification process incurs one additional exponentiation, which is expensive. The use of c in the protocol thwarts the small subgroup attack at a much lower cost of one multiplication, because the element y^c is a random element of $\langle g \rangle$ provided that y is a random element of $\mathbb{Z}_p{}^*$. Note that a similar countermeasure against small subgroup attacks can be found in the design of the MQV key agreement scheme (see [18]). The small subgroup attack is one of many possible attacks prevented by the zero-knowledge property of the protocol, which will be formally proved later.

3 Security Analysis

3.1 Zero-Knowledge Properties of the Scheme

3.1.1 Perfect Zero-Knowledge for Honest Verifier

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). If A rejects, we denote the transcript as $(y_1, h_1, \perp)$, where "$\perp$" is a "reject" symbol.

In the case where the verifier is honest (i.e., the verifier follows the protocol), the real transcript produced by the prover and the verifier during a session can be simulated by the verifier without participating the session. The distribution of simulated transcripts is identical to the distribution of real transcripts. Thus, we prove that Protocol 2.2 is perfect zero-knowledge for an honest verifier.

Theorem 3.1. *Protocol 2.2 is perfect zero-knowledge for an honest verifier.*

Proof. The set $\mathcal{T}$ of real transcripts obtained by a prover and an honest verifier consists of all transcripts T having the following form:

$$\begin{aligned} T &= (y_1, h_1, y_2) \\ &= (g^r, h(v^{rc}||g^r), (g^r)^{ac}) \\ &= (g^r, h(g^{arc}||g^r), g^{arc}) \end{aligned}$$

where r is chosen by the verifier uniformly at random from $\mathbb{Z}_q$.

The set $\hat{\mathcal{T}}$ of simulated transcripts can be constructed by the verifier as follows. The verifier chooses r uniformly at random from $\mathbb{Z}_q$, and computes the simulated transcript $\hat{T}$, using g, v, and $h(\cdot)$, as follows:

$$\begin{aligned} \hat{T} &= (g^r, h(v^{rc}||g^r), v^{rc}) \\ &= (g^r, h(g^{arc}||g^r), g^{arc}). \end{aligned}$$

The random numbers r in both T and $\hat{T}$ have identical probability distributions. Therefore $\mathcal{T}$ and $\hat{\mathcal{T}}$ have identical probability distributions, and the protocol is perfect zero-knowledge for an honest verifier. □

3.1.2 Statistical Zero-Knowledge with Dishonest Verifier

Now we consider the case in which a verifier might be dishonest. A dishonest verifier may generate messages not following the protocol, e.g., he may produce a pair (y_1, h_1) without first choosing r_1, or perhaps he chooses r_1 with a non-uniform probability distribution. We show that the protocol is statistically zero-knowledge if the Knowledge-of-Exponent Assumption (KEA) holds and the hash function h is a random oracle.

KEA is a non-standard cryptographic assumption first proposed by Damgård in [7]. Its application can be found in various papers, e.g., [4, 16, 17]. 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 the 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 3.1. *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" $\boldsymbol{E}$, which takes the same input and outputs the pair (d, d') along with an exponent r such that for sufficiently large k,*

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

for any polynomials $Q()$.

Remark 3.1. Our formal definition of KEA is based on [4], but it is a slight extension in that we explicitly mandate that $d \in \langle g \rangle$. Without this condition, the assumption no longer holds. For example, suppose $\mathbf{A}$ chooses r and outputs $(d = -g^r, d' = (g^a)^r)$. If a is even (and there is a 50% chance that it is, since a is chosen randomly), then $d^a = d'$, but no algorithm can find r such that $g^r = d, (g^a)^r = d'$ because $d \notin \langle g \rangle$. This is actually the same idea used in the small subgroup attack that we discussed in Section 2.

Now, we consider a general polynomial-time verifier. This verifier can generate challenges (valid or invalid) in whatever manner he wishes. For example, the verifier might sometimes generate valid challenges (y_1, h_1) by first computing $y_1 = g^r$ and then computing $h_1 = h(v^r \bmod p)$; at other times, the verifier might generate challenges (y_1, h_1) by some other poly-time method. Informally, the KEA allows us to conclude that there is negligible probability that the verifier generates a valid challenge without "knowing" the value of r.

Theorem 3.2. *If the KEA holds and $h(\cdot)$ is a random oracle, then the protocol is statistical zero-knowledge for an arbitrary verifier.*

Proof. Since all computations take place the in group $\mathbb{Z}_p$, we assume that $y_1 \in \mathbb{Z}_p$. Obviously if $y_1 = 0$, the verifier knows the prover will reject. Therefore we only consider the case that $y_1 \in \mathbb{Z}_p^*$.

Recall that $c = (p-1)/q$. As mentioned earlier, for any $y_1 \in {\mathbb{Z}_p}^*$, it holds that ${y_1}^c \in \langle g \rangle$. From the KEA, we know for random a, if $\mathbf{A}(g, g^a)$ returns $({y_1}^c, y_2)$, then there exists an extractor $\mathbf{E}(g, g^a)$ that returns $({y_1}^c, y_2, r)$ such that, for sufficiently large k, it holds that

$$\Pr[y_2 = ({y_1}^c)^a \text{ and } g^r \neq {y_1}^c] < \frac{1}{Q(k)} \tag{1}$$

for any polynomial $Q()$.

Now B simulates transcripts as follows: first he generates the challenge pair (y_1, h_1) the same way as he would in a real session, and then he completes the transcript as follows:

1. Suppose h_1 is not generated by querying $h(y_2||y_1)$ for some y_2. Then B outputs the transcript
$$(y_1, h_1, \perp).$$

2. Suppose h_1 is a reply of $h(y_2||y_1)$ for some input y_2. This means that B generates (y_1, y_2) before invoking the random oracle h. Then B computes the pair $({y_1}^c, y_2)$, and computes r using the corresponding extractor $\mathbf{E}$: $\mathbf{E}(g, v) = ({y_1}^c, y_2, r)$. If $({y_1}^c, y_2) = (g^r, v^r)$, then B outputs the transcript
$$(y_1, h_1, y_2);$$
otherwise, B outputs the transcript
$$(y_1, h_1, \perp).$$

From the description of the protocol and the process that B uses to simulate transcripts, we see that a difference between the probability distributions of real transcripts $\mathcal{T} = \{T\}$ and simulated transcripts $\hat{\mathcal{T}} = \{\hat{T}\}$ can happen only when the challenge pair (y_1, h_1) is valid, i.e., when $h_1 = h({y_1}^{ac}||y_1)$. In this situation, the real transcript $T = (y_1, h_1, {y_1}^{ac})$, while the simulated transcript $\hat{T} = (y_1, h_1, {y_1}^{ac})$ or $(y_1, h_1, \perp)$. In the following three cases, the simulated transcript will be $(y_1, h_1 = h({y_1}^{ac}||y_1), \perp)$:

1. h_1 is not the reply of a query $h(y_2||y_1)$ for some y_2, but $h_1 = h({y_1}^{rc}||y_1)$. We denote such an event by e_1. Obviously $\Pr[e_1] = 1/2^k$.

2. h_1 is the reply of a query $h(y_2||y_1)$ for some y_2 and $h(y_2||y_1) = h({y_1}^{ac}||y_1)$, but $y_2 \neq {y_1}^{ac}$. We denote such an event by e_2. Obviously $\Pr[e_2] = 1/2^k$.

3. h_1 is the reply of a query $h(y_2||y_1)$ and $y_2 = ({y_1}^c)^a$, but the extractor fails to find r such that $g^r = {y_1}^c$. We denote such an events by e_3. From (1), we know that $\Pr[e_3] < 1/Q(k)$.

Now, the statistical distance d between the distributions on real transcripts and simulated transcripts is computed as follows:

$$\begin{aligned} d &= \Pr[e_1 \text{ or } e_2 \text{ or } e_3] \\ &< \Pr[e_1] + \Pr[e_2] + \Pr[e_3] \\ &< 1/Q(k) + 1/2^{k-1}. \end{aligned}$$

Hence, the statistical distance between the probability distributions on $\mathcal{T}$ and $\hat{\mathcal{T}}$ is negligible, and the protocol is statistical zero-knowledge. □

Remark 3.2. The above analysis actually shows the protocol has a property stronger than zero-knowledge: when the challenge (y_1, h_1) is constructed by the verifier, he knows (with overwhelming probability) what an honest prover will reply. This implies zero-knowledge. On the other hand, zero-knowledge does not necessarily imply that the verifier knows the reply in advance.

3.1.3 Zero-knowledge in Concurrent and Reset Settings

The zero-knowledge property of our protocol is preserved under the three types of composition of sessions of the protocol; namely, *sequential composition*, *parallel composition* and *concurrent composition* (see, e.g., [14]). In sequential composition, the protocol is invoked many times, where each invocation follows the termination of the previous one. Every zero-knowledge protocol is zero-knowledge under sequential composition ([14]).

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 ([14]). In our protocol, however, the prover is stateless and deterministic. When he receives a challenge, the prover replies immediately and the session ends, and then he proceeds with the next session. Since each reply is determined only by the corresponding challenge, parallel composition of sessions of our protocol are equivalent to sequential compositions. Therefore our protocol maintains zero-knowledge properties under parallel composition.

Concurrent composition generalizes both sequential and parallel composition. In this setting, many instances of the protocol are invoked at arbitrary times and proceed at an arbitrary pace. Again, for our protocol, any concurrent composition of sessions of the protocol is equivalent to a sequential composition, because the prover is stateless and deterministic.

Our protocol also preserves zero-knowledge under the reset setting, where the adversary can reset the internal state of the prover. Again, this follows immediately because the prover is stateless and deterministic.

3.2 Security of the Scheme

In this section, we give a proof of security of our scheme, based on the intractability of the Computational Diffie-Hellman Problem (CDH). The Computational Diffie-Hellman Problem is to compute the value of g^{ab}, given g, g^a and g^b. The Computational Diffie-Hellman Assumption (denoted by CDHA), is defined formally as follows.

Definition 3.2. *The* Computational Diffie-Hellman Assumption *is as follows: 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 $Q(\cdot)$, it holds that*

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

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

3.2.1 Security under Sequential, Concurrent and Reset Attacks

The goal of an adversary is to be able to impersonate a prover. The following result relates what it means for an impersonation (for Protocol 2.2) to be considered successful, under a sequential, concurrent or reset attack, to a computational problem similar to the Diffie-Hellman problem.

Theorem 3.3. *Suppose an adversary is impersonating a prover with public key $v = g^a$ using a sequential, concurrent or reset attack. Then the adversary can impersonate the prover with non-negligible probability if and only if the adversary can compute g^{ab}, when he 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 $Q_1(\cdot)$, such that for all sufficiently 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 $k = \log_2 q$.

Proof. In Protocol 2.2, a prover has a public key $v = g^a$. If the challenge value $y_1 = g^b$, then the adversary must compute g^{abc} to successfully impersonate the prover, where $c = (p-1)/q$. Recall that we are assuming that $\gcd(c, q) = 1$, so $c' = c^{-1} \bmod q$ exists. Therefore $g^{ab} = (g^{abc})^{c'}$, and computing g^{ab} is polynomially equivalent to computing g^{abc}. □

We have given a result stating what it means for an impersonation to be successful. Now we can define what it means for an identification scheme to be secure.

Definition 3.3. *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.*

Specifically, in the case of concurrent or reset attack, the security definition requires that for any polynomial $Q()$, and for all sufficiently large k, it holds that

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

where $v = g^a$, a is randomly chosen from $\mathbb{Z}_q$ and $\mathbf{A}_v$ is as defined in Theorem 3.3.

Now, we prove that Protocol 2.2 is secure against sequential, concurrent and reset attacks under the CDH and KEA Assumptions.

Theorem 3.4. *Protocol 2.2 is secure against concurrent and reset attack in the random oracle model under the CDH and KEA Assumptions.*

Proof. Suppose that the protocol is not secure against concurrent and reset attack. Because the KEA holds, we know from Theorem 3.2 that the protocol is zero-knowledge in the random oracle model. Hence, the adversary can impersonate a prover without taking part in or observing any previous sessions of the protocol.

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,

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

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 3.3. Therefore, given a pair of elements g^a, g^b, where a, b are randomly chosen from $\mathbb{Z}_q$, the probability that the adversary can compute g^{ab} is at least

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

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

3.2.2 Security Under Active-intruder Attacks

Next we prove our protocol is secure against active-intruder attacks.

Theorem 3.5. *Protocol 2.2 is secure against active-intruder attacks in the random oracle model under the CDH and KEA Assumptions.*

Proof. We need to prove that, after the adversary is active in a session S, the verifier will reject. We use a three bit string to indicate which of the three items y_1, h_1, y_2 are not faithfully relayed, respectively. A "1" indicates that the corresponding item is altered, and a "0" indicates that it is not altered. Let y_1', h_1', y_2' represent altered items. There are a total of seven cases to examine:

case 001 Here, the verifier will reject because $y_2' \neq y_2 = {y_1}^{ac} = v^{rc}$.

case 010 Here, the prover will reject because $h_1' \neq h_1 = h(v^{rc})||y_1) = h({y_1}^{ac}||y_1)$. Hence the verifier will also reject.

case 011 Here, the prover will reject as in the previous case. Moreover, the adversary knows that the prover will reject. Then, after the prover rejects, the adversary sends a new value y_2' to the verifier. If the adversary can make the verifier accept by sending y_2', he could do the same thing without interacting with the prover at all in the session S. This implies that the adversary can carry out a successful concurrent attack, contradicting Theorem 3.4.

case 100 In this case, there is only a negligible probability that the adversary can find $y_1' \neq y_1$ such that $h({y_1'}^{ac}||y_1') = h_1$. This is because the hash function h is a random oracle. Therefore there is an overwhelming probability that the prover with reject, and the verifier will also reject.

case 101 As in the previous case, there is an overwhelming probability that the prover will reject. If the adversary can then cause the verifier to accept by sending a new value y_2', he can do the same thing without interacting with the prover in the session S. This implies a successful concurrent attack, contradicting Theorem 3.4.

case 110 In this case, if the prover rejects, then the verifier will reject too. So, suppose (y_1', h_1') is accepted by the prover. (y_1', h_1') is either generated by the adversary or copied from a previous session. In the former case, the adversary can predict (with overwhelming probability) what the prover will reply; in the latter case, the adversary also knows what the reply is from the previous session. Therefore, the adversary can make the verifier accept without actually interacting with the prover in the session S. That implies that a concurrent attack is successful, which contradicts Theorem 3.4.

case 111 If the prover rejects, then the adversary can make the verifier accept without interacting with the prover in the session S, which means a concurrent attack is successful. If the prover accepts, then as analyzed in the previous case, it also means that a successful concurrent attack has occurred. In any case, Theorem 3.4 is contradicted.

Summarizing, we conclude that if the adversary is active in a session S, then the verifier will reject with overwhelming probability in the session S. □

4 Performance

We compare the performance of our scheme with the Schnorr Scheme, which is one of the most efficient identification schemes known. Assume, for both protocols, that $|p| = 1024$ bits, $|q| = 160$ bits, and in our protocol $|h()| = 160$ bits. The total message length in the Schnorr Scheme is 1344 bits, and the message length in our protocol is 2208 bits. If both protocols use a hash function to compress the messages (in Schnorr scheme, by hashing the 1024-bit commitment, and in our protocol by hashing the 1024-bit response), the total message length is 480 bits and 1344 bits respectively.

For computational complexity, we compare the number of exponentiations required. In the Schnorr scheme, there are two exponentiations for the verifier and one exponentiation for the prover, all with a 160-bit exponent. The computational requirements are the same in our scheme. However, in the Schnorr Scheme, the exponentiation for the prover can be precomputed. This is an advantage in the smartcard setting, where the prover is a smartcard whose computing power is very limited.

One advantage of our scheme is that a session consists of two flows, which is preferable to the three flows required in the Schnorr Scheme. In most cases the prover will initiate a session by declaring its identity; however, for two-flow identification, the "declaration" message is not bound to a specific session. In some cases, e.g., when the prover needs to identify itself independently to a group of verifiers, it can broadcast its identity and then each subsequent identification session still has two flows. Moreover, the prover in a two-flow scheme can be implemented as a stateless server. This is simpler than the structure of the three-flow identification protocols, which must be stateful and which require a built-in random number generator.

5 Conclusion

In this paper, we proposed a new two-flow zero-knowledge identification scheme. Based on the CDH and KEA assumptions, it is provably secure in random oracle model under the strongest attack model to date, which incorporates sequential, concurrent, reset and active-intruder attacks. While it achieves all these security properties, it is still very efficient. The total message length and computation required in a session are reasonably close to those of the Schnorr Scheme, which is among the most efficient known identification schemes.

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 the Thirtieth Annual ACM Symposium on Theory of Computing*, ACM Press, New York, 1998, pp. 419–428.

[2] M. Bellare, S. Goldwasser and S. Micali. Identification protocols secure against reset attacks, *Lecture Notes in Computer Science* **2045** (2001), 495–511 (EUROCRYPT 2001 Proceedings).

[3] M. Bellare and A. Palacio. GQ and Schnorr identification schemes: proofs of security against impersonation under active and concurrent attacks, *Lecture Notes in Computer Science* **2442** (2002), 162–177 (CRYPTO 2002 Proceedings).

[4] M. Bellare and A. Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. *Lecture Notes in Computer Science* **3152** (2004), 273–289 (CRYPTO 2004 Proceedings).

[5] M. Bellare, D. Pointcheval and P. Rogaway. Authenticated key exchange secure against dictionary attacks. *Lecture Notes in Computer Science* **1807** (2000), 139–155 (EUROCRYPT 2000 Proceedings).

[6] M. Bellare and P. Rogaway. Entity authentication and key distribution. *Lecture Notes in Computer Science* **773** (1994), 232–249 (CRYPTO '93 Proceedings).

[7] I. Damgård. Towards practical public key systems secure against chosen ciphertext attacks, *Lecture Notes in Computer Science* **576** (1992), 445–456 (CRYPTO '91 Proceedings).

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

[9] U. Feige, A. Fiat, and A. Shamir. Zero knowledge proofs of identity. *J. Cryptology* **1** (1988), 77–94.

[10] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. Advances in Cryptology, *Lecture Notes in Computer Science* **263** (1987), 186–194 (CRYPTO '86 Proceedings).

[11] M. Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors, *Lecture Notes in Computer Science* **3621** (2005), 152–168 (CRYPTO '05 Proceedings).

[12] J. Furukawa, K. Kurosawa, and H. Imai. An efficient compiler from Σ-protocol to 2-move deniable zero-knowledge, *Lecture Notes in Computer Science* **4052** (2006), 46–57 (ICALP 2006 Proceedings).

[13] O. Goldreich. *Foundations of Cryptography I: Basic Tools*. Cambridge University Press, 2001

[14] O. Goldreich. Zero-knowledge twenty years after its invention, unpublished manuscript available from `http://www.wisdom.weizmann.ac.il/~oded/zk-tut02.html`

[15] L. Guillou and J.-J. Quisquater. A "paradoxical" identity-based signature scheme resulting from zero-knowledge, *Lecture Notes in Computer Science* **403** (1990), 216–231 (CRYPTO '88 Proceedings).

[16] S. Hada and T. Tanaka. On the existence of 3-round zero-knowledge protocols, *Lecture Notes in Computer Science* **1462** (1998), 408–423 (CRYPTO '98 Proceedings).

[17] H. Krawczyk. HMQV: A high-performance secure Diffe-Hellman protocol, *Lecture Notes in Computer Science* **3621** (2005), 546–466 (CRYPTO 2005 Proceedings).

[18] L. Law, A.J. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography* **28** (2003), 119–134.

[19] C.H. Lim and P.J. Lee, A key recovery attack on discrete log-based schemes using a prime order subgroup, *Lecture Notes in Computer Science* **1294** (1997), 249–263 (CRYPTO '97 Proceedings).

[20] A.J. Menezes, P.C. van Oorschot and S.A. Vanstone. *Handbook of Applied Cryptography*, CRC Press, 1996.

[21] T. Okamoto. Provably secure and practical identitication schemes and corresponding signature schemes, *Lecture Notes in Computer Science* **740** (1993), 31–53 (CRYPTO '92 Proceedings).

[22] R. Pass. On deniability in the common reference string and random oracle model. *Lecture Notes in Computer Science* **2729** (2003), 316–337 (CRYPTO 2003 Proceedings).

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

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

A Active-intruder Attacks

We use simple figures and notations to illustrate several identification protocols and corresponding active-intruder attacks on them. Among these notations, r is a random number chosen by A and e is a random number chosen by B. All computations take place in a relevant group.

A.1 Attack on the Schnorr Scheme

Schnorr Scheme: g is a public parameter, a is secret key and $v = g^{-a}$ is public key.

$$\begin{array}{ccc} A & \xrightarrow{x=g^r} & B \\ A & \xleftarrow{e} & B \\ A & \xrightarrow{y=ae+r} & B \end{array}$$

B verifies that

$$g^y v^e = g^{ae+r} v^e = g^{ae+r}(g^{-a})^e = g^r = x$$

and accepts.

Attack:

$$\begin{array}{rcl} A \xrightarrow{g^r} & O & \xrightarrow{x=g^{2r}} B \\ A \xleftarrow{e/2} & O & \xleftarrow{e} B \\ A \xrightarrow{ae/2+r} & O & \xrightarrow{y=ae+2r} B \end{array}$$

B verifies that

$$g^y v^e = g^{ae+2r} v^e = g^{ae+2r}(g^{-a})^e = g^{2r} = x$$

and accepts.

A.2 Attack on the Fiat-Shamir Scheme

Fiat-Shamir scheme: a is secret key, $v = a^2$ is public key.

$$\begin{array}{ccc} A & \xrightarrow{x=r^2} & B \\ A & \xleftarrow{e\in\{0,1\}} & B \\ A & \xrightarrow{y=ra^e} & B \end{array}$$

B verifies that

$$y^2 = r^2a^{2e} = xv^e$$

and accepts.

Attack:

$$\begin{array}{rcl} A \xrightarrow{r^2} & O & \xrightarrow{x=k^2r^2} B \\ A \xleftarrow{e} & O & \xleftarrow{e\in\{0,1\}} B \\ A \xrightarrow{ra^e} & O & \xrightarrow{y=kra^e} B \end{array}$$

B verifies that

$$y^2 = k^2r^2a^{2e} = xv^e$$

and accepts.

A.3 Attack on the Okamoto Scheme

Okamoto scheme: a_1, a_2 are secret keys, $v = {g_1}^{-a_1}{g_2}^{-a_2}$.

$$\begin{array}{ccc} A & \xrightarrow{x={g_1}^{r_1}{g_2}^{r_2}} & B \\ A & \xleftarrow{e} & B \\ A & \xrightarrow{y_1=a_1e+r_1, y_2=a_2e+r_2} & B \end{array}$$

B verifies

$${g_1}^{y_1}{g_2}^{y_2}v^e = {g_1}^{a_1e+r_1}{g_2}^{a_2e+r_2}({g_1}^{-a_1}{g_2}^{-a_2})^e = {g_1}^{r_1}{g_2}^{r_2} = x$$

and accepts.

Attack:

$$\begin{array}{rcl} A \xrightarrow{{g_1}^{r_1}{g_2}^{r_2}} & O & \xrightarrow{x={g_1}^{2r_1}{g_2}^{2r_2}} B \\ A \xleftarrow{e/2} & O & \xleftarrow{e} B \\ A \xrightarrow{ae/2+r_1, ae/2+r_2} & O & \xrightarrow{y_1=ae+2r_1, y_2=ae+2r_2} B \end{array}$$

B verifies

$${g_1}^{y_1}{g_2}^{y_2}v^e = {g_1}^{a_1e+2r_1}{g_2}^{a_2e+2r_2}({g_1}^{-a_1}{g_2}^{-a_2})^e = {g_1}^{2r_1}{g_2}^{2r_2} = x$$

and accepts.

A.4 Attack on the GQ Scheme

GQ scheme: a is secret key, v is public key, $v = a^{-b}$, b is a public parameter.

$$\begin{array}{ccc} A & \xrightarrow{x=r^b} & B \\ A & \xleftarrow{e} & B \\ A & \xrightarrow{y=ra^e} & B \end{array}$$

B verifies

$$v^e y^b = a^{-be} r^b a^{be} = r^b = x$$

and accepts.

Attack:

$$\begin{array}{ccc} A \xrightarrow{r^b} & O & \xrightarrow{x=2^b r^b} B \\ A \xleftarrow{e} & O & \xleftarrow{e} B \\ A \xrightarrow{ra^e} & O & \xrightarrow{y=2ra^e} B \end{array}$$

B verifies

$$v^e y^b = a^{-be} 2^b r^b a^{be} = 2^b r^b = x$$

and accepts.