Efficient Identification Protocols and the Knowledge-of-Exponent Assumption

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

Abstract

In this paper, we propose an extremely simple identification protocol and prove its security using the Knowledge-of-Exponent Assumption (KEA). We discuss the applicability of KEA in various protocol settings as well. Recently, doubts have been raised about applying KEA in some protocols where an adversary has auxiliary inputs. However, we suggest that KEA is applicable in these cases. We present two variants of KEA, Generalized KEA (GKEA) and Auxiliary-Input KEA (AI-KEA), to clarify the proper use of KEA.

1 Introduction

An identification protocol or scheme is an interactive protocol between a *prover* and a *verifier*. The prover tries to identify herself to the verifier by demonstrating knowledge of a certain key associated with the prover. 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. 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 [11], the Schnorr Scheme [16], the Feige-Fiat-Shamir (FFS) Scheme [10], the Guillou-Quisquater (GQ) Scheme [12] and the Okamoto Scheme [15]. In general, these zero-knowledge proof of knowledge type identification schemes are not secure in the strongest attack model [19].

In [19], Stinson and Wu propose an efficient 2-flow zero-knowledge identification protocol. The protocol is secure in the strongest security model considered to date, and it is very efficient as well. 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. The security of the protocol is proved using the knowledge-of-exponent assumption (KEA) in the random oracle model.

KEA was first proposed by Damgård in [8]. 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 (x, x^a) is to first compute $x = g^r$ for some r, and then compute $x^a = (g^a)^r$. A formal definition is as follows.

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

Definition 1.1. *The* Knowledge-of-Exponent Assumption *is as follows: Let g be a generator of a prime order group $\langle g \rangle$. $k = \log |\langle g \rangle|$ is the security parameter. For any probabilistic polynomial-time (PPT) algorithm $\boldsymbol{A}$ that takes as input g and g^a, where a is an unknown randomly chosen integer, and which produces as output a pair of the form $(x, y), x \in \langle g \rangle$, there exists a PPT "extractor" $\boldsymbol{E}$, which takes the same input and outputs the pair (x, y) along with an exponent r such that for sufficiently large k,*

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

for any polynomial Q.

Our formal definition of KEA is based on [3], but it is a slight extension in that we explicitly mandate that $x \in \langle g \rangle$. Without this condition, the assumption no longer holds. For example, suppose **A** chooses r and outputs $(x = -g^r, y = (g^a)^r)$. If a is even (and there is a 50% chance that it is, since a is chosen randomly), then $x^a = y$, but no PPT algorithm can find r such that $g^r = x, (g^a)^r = y$ because $x \notin \langle g \rangle$.

KEA has been used in a number of applications. In [13], Hada and Tanaka used KEA (in the name DA-1) to prove the existence of 3-round zero-knowledge proofs. In [3], Bellare and Palacia revisited KEA (in the name KEA1) and rectified the proof of [13], but KEA is not found false. In [4], Bellare and Palacia used KEA (in the name DHK0) to prove the paintext-aware property of DEG. In [9], Dent used KEA (in the name DHK) to prove the paintext-aware property of Cramer-Shoup scheme [7]. In [14], Krawcyzk used KEA (in the name KEA1) to prove the security of HMQV, a high-performance key agreement protocol. In [19] and [20], Stinson and Wu used KEA to prove the security of an efficient identification protocol and the security of a key agreement protocol respectively.

In [21], Yao et al raised a question about applying KEA in some protocols such as [4], [9], [14]. They argued that in these protocols, besides the pair (g, g^a), the adversary can get auxiliary inputs, i.e., pairs of the form $(x_i, {x_i}^a)$ by observing the protocol executing between other honest parties, then the adversary can compute a new pair constructed from these auxiliary inputs, say $(x = {x_1}^{r_1} {x_2}^{r_2}, y = {y_1}^{r_1} {y_2}^{r_2})$. In this case, the adversary outputs (x, x^a), but it is perhaps not reasonable to assume the adversary "knows" the exponent r of x, and therefore KEA is not applicable in such cases.

Our work in this paper is two-fold. First, we propose an identification protocol which achieves the same security as the Stinson-Wu protocol in [19], but is more efficient. Second, we suggest that the argument in [21] does not invalidate the application of KEA in [4], [9], [14].

The remainder of the paper is organized as follows. In Section 2, we present our new identification protocol and prove its security. In Section 3, we discuss the applicability of KEA, and we conclude the paper in Section 4.

2 A Simple Identification Protocol

In this section, we propose an extremely simple identification protocol and prove its security based on KEA.

Input.	Security parameters k and k', which are positive integers. The parameter k' should be polynomial in k.
1.	The *TA* chooses a large prime p such that $p-1$ is divisible by another large prime q, where $\log_2 p \approx k'$ and $\log_2 q \approx k$.
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 $h : {\mathbb{Z}_p}^* \rightarrow \{0,1\}^k$.
5.	For each potential prover **P**, the *TA* chooses a private key a uniformly from $\mathbb{Z}_q$ at random, computes $v = g^a \mod p$, and issues a certificate to **P** certifying that v is indeed **P**'s public key.

Figure 1: Identification Scheme Setup

2.1 Initial Setup

The initial setup for our scheme is described in Figure 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 **P** tries to convince the verifier **V** of her identity. **V** "accepts" only if **P** responds to **V**'s challenge in an appropriate way. The steps in a session of our scheme are summarized in Figure 2.

In the following, we omit the operation "mod p" to simplify the notation. The message flows can be depicted as follows:

$$\mathbf{P} \quad \xleftarrow{x=g^r} \quad \mathbf{V}$$
$$\mathbf{P} \quad \xrightarrow{y=h(x^a)} \quad \mathbf{V}$$

Remark. 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 verification that $x^q \mod p \neq 1$ is unnecessary; it would suffice to verify that x is a point on E.

2.3 Security proof

The following attacks against an identification scheme have been identified in the literature, capturing the capability of the adversary in different settings: sequential attacks, concurrent attacks, active-intruder attacks, and reset attacks. For a detailed description of these attacks, see [19].

In this section, we prove the identification protocol is secure against these attacks if KEA holds and the hash function $h()$ is a random oracle.

The security of an identification protocol is defined as follows:

Definition 2.1. *An identification protocol is secure if any PPT adversary cannot impersonate a non-negligible portion of all possible provers with non-negligible probability.*

1. **V** chooses $r \in \mathbb{Z}_q$ uniformly at random and computes

$$x = g^r \bmod p.$$

Then **V** sends x to **P**.

2. After receiving x, **P** rejects and stops if $x^q \bmod p \neq 1$; otherwise **P** computes

$$y = h(x^a \bmod p).$$

and sends y to **V**; otherwise **P** rejects and stops.

3. After receiving y, **V** verifies y. If $y_2 = h(v^r \bmod p)$, then **V** accepts; otherwise, **V** rejects.

Figure 2: Identification scheme description

The attack model we use is a combination of active-intruder attacks and reset attacks. Note that active-intruder attacks cover concurrent attacks, which cover sequential attacks in turn.

Sequential Attacks

In sequential attacks, the adversary can act as a verifier and interact with the prover in various sessions before the adversary tries to impersonate the prover. The sessions between the adversary and the prover are carried out sequentially.

We have the following result.

Theorem 2.2. *Let a be the private key of a prover $\boldsymbol{P}$, and let x be a random challenge. If the KEA assumption holds. Then for sufficiently large k, it is infeasible for any PPT $\boldsymbol{A}$, given Q_1 accesses to $\boldsymbol{P}$ and Q_2 accesses to $h()$ (where Q_1, Q_2 are polynomials in k), to compute $h(x^a)$ for a non-negligible fraction of all possible $\boldsymbol{P}$.*

Proof. A simulator **S** runs the protocol as a prover with the adversary as a verifier in a simulation. **S** maintains a hash table T consisting of three columns x, y, h. Let X, Y, H be the set of x, y, h values respectively. T is initially empty. The algorithm used by **S** is described in Algorithm 1.

Roughly speaking, **S** simulates a random oracle. **S** maintains a hash table, so that when **A** sends a challenge x, it receives the same answer as it would when it queries the hash value of x^a, and vice versa. When **A** sends a challenge x, **S** relies on the extractor **E** to find out if **A** has queried the hash of x^a, and vice versa, so as not to give inconsistent answers. **E** extracts an r for each pair of (x, y) where x is a challenge from **A** and y is a hash input from **A**. Supposing that **A** has made m challenges $x_1, \cdots, x_m$ and n hash queries $h(y_1), \cdots, h(y_n)$. If **A** makes a new challenge x_{m+1}, then **E** will output n triples $(x_{m+1}, y_j, r_{m+1,j}), 1 \leq j \leq n$. Similarly, if **A** makes a new hash query $h(y_{n+1})$, then **E** will output m triples $(x_i, y_{n+1}, r_{i,n+1}), 1 \leq i \leq m$. For each triple (x, y, r), by checking if $x = g^r$ and $y = v^r$, **S** decides if $y = x^a$.

Assuming **A** makes a polynomial number (in k) Q_1 of challenges and polynomial number Q_2 of hash queries, it is clear that **S** runs in polynomial time.

We compare the probability distributions of transcripts **A** receives in the simulation and the transcripts **A** receives in real transactions with **P**. The transcript consists of all challenges and their

Algorithm 1: **S**

```
repeat
    receive a challenge x or a hash query on y from A;
    if challenge x then
        if x ∈ X then
            return h in the same row in T
        else
            for each pair (x, y) ∈ {x} × Y do
                receive r from extractor E for the pair (x, y)
                if x = g^r and y = v^r then
                    fill x in the same row as y in T
                    return h to A
            add x to a new row of T
            fill in a random h in that row
            return h to A
    else
        if y ∈ Y then
            return h in the same row in T
        else
            for each pair (x, y) ∈ X × {y} do
                receive r from extractor E for the pair (x, y)
                if x = g^r and y = v^r then
                    fill y in the same row as x in T
                    return h from the same row to A
            add x to a new row in T
            fill in a random h in that row
            return h to A
until ;
```

answers, as well as all hash queries and their answers. A difference between the two probability distributions happens in two cases:

1. **A** sends a challenge x after it has queried the hash of x^a, and receives a reply from **S** which is different from the hash of x^a.

2. **A** queries the hash of y after it has sent a challenge x where $y = x^a$, and receives a hash of y which is different from the reply for x from **S**.

Given a polynomial number Q_1 of challenges and polynomial number Q_2 of hash queries, there are at most Q_1Q_2 pairs (x, y) considered in the algorithm. For each pair which satisfies $y = x^a$, by KEA, the above inconsistent event happens with negligible probability (i.e., $\Pr \leq 1/Q$ for any polynomial Q for k large enough). The statistical difference between the two distributions is at most $\Pr \times Q_1Q_2 \leq Q_1Q_2/Q$, which is negligible. Therefore, the two distributions are statistically indistinguishable. □

Concurrent Attacks

In sequential attacks, the adversary can only execute multiple sessions sequentially. In concurrent attacks, many instances of the protocol can be invoked by the adversary (as verifiers) at arbitrary times and proceed at an arbitrary pace. The adversary tries to gain some knowledge about the prover's private key in the attacks, then the adversary uses it to impersonate the prover. Concurrent attacks cover sequential attacks. Note that protocols secure under sequential attacks may be insecure under concurrent attacks.

In our protocol, the prover is stateless and deterministic. The output of the prover only depends on the input challenge. When the prover receives a challenge, the prover replies immediately and the session ends, and then she proceeds with the next session. Therefore, concurrent compositions of sessions of our protocol are equivalent to sequential compositions, thus our protocol remains secure under concurrent attacks.

Active-intruder Attack

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 other words, the active-intruder attack is a "man-in-the-middle" type of attack.

In such a setting, it is not straightforward to define what it means for an attack to be successful. There are three approaches to handling this definitional issue. The first is based on the idea of *matching conversations* [6] by Bellare and Rogaway. A somewhat more formal notion is that of *matching session IDs* [5, 1] by Bellare et al. An other approach is due to Stinson [17, §11.1] and is described using the idea of an active intruder. Informally, an adversary is an *active intruder* in a session if he alters, injects, drops, and/or diverts at least one message in the given session. (If the adversary simply relays messages between the prover and verifier, then the adversary is not an active intruder.) Based on this notion, we define a successful *active-intruder attack* as follows:

Definition 2.3. *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 remark that the notions of matching conversations and an inactive adversary are equivalent, in that matching conversations take place in a session if and only if there is no active adversary in that session.

We formalize a property of random oracles which we need in our proof.

Property 2.4. *Suppose that $h : X \to Y$ is a random oracle, and let $x_1, \cdots, x_2 \in X$. Suppose that the values $y_i = h(x_i)$ have been determined for $1 \leq i \leq l$. Then $\Pr[h(x) = y | x \notin \{x_1, \cdots, x_l\}] = \frac{1}{|Y|}$ for all $y \in Y$. Note that the probability is computed over all possible functions $h()$.*

This property is called the *independence property* of a random oracle in [18].

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

Theorem 2.5. *The protocol presented in Figure 2 is secure against polynomial-time active-intruder attacks in the random oracle model if it is secure under concurrent attacks.*

Proof. We need to prove that, after the adversary **A** is active in a session S, the verifier will reject. There are a total of three cases to examine:

1. x is altered and y is intact. In this case, the prover receives $x' \neq x$ and sends back $y = h(x'^a)$. Since $x' \neq x$, with overwhelming probability $h(x^a) \neq h(x'^a)$. Then the verifier rejects with overwhelming probability.

2. x is intact and y is altered. In this case, the adversary changes $y = h(x^a)$ to $y \neq h(x^a)$. The verifier will reject.

3. Both x and y are altered. In this case, **A** receives x from **V**, sends $x' \neq x$ to **P**, then receives $y' = h(x'^a)$ from P, and sends $y \neq y'$ to **V**. Since the hash function $h()$ is assumed to be a random oracle, by the independence property of $h()$, $h(x'^a)$ for any $x' \neq x$ does not help **A** to compute $h(x^a)$. If **A** can make **V** accept, he can do that without receiving $h(x'^a)$ from **P**. That means a successful concurrent attack.

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. □

Reset Attacks

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.

Our protocol is secure under reset attacks. This follows immediately because the prover is stateless.

Security Summary

The combination of active-intruder and reset attacks is the strongest attack model that has been considered for identification schemes to date. Our protocol is secure under this attack model.

2.4 Performance

We compare the performance of our scheme with the Stinson-Wu (SW) scheme in [19]. Assume, for all protocols, that $k' = 1024$ and $k = 160$. The total message length is 2208 bits in SW scheme, and 1184 bits in our scheme. Note that a hash function can be used to compress the messages in SW scheme (by hashing the 1024-bit response to 160 bits) so its message length can be reduced to 1344 bits.

In view of the number of exponentiations, the computational complexity of the two schemes are the same. Both the prover and verifier need two exponentiations, all with a 160-bit exponent. In the elliptic curve setting, provers in both scheme are required to perform only one "exponentiation" (i.e., a scalar multiple of a point on the elliptic curve).

Another advantage of the new scheme is its simple security proof, which partially results from the simple message flow of the scheme.

3 Applicability of KEA

KEA does not explicitly address the settings where multiple parties execute multiple sessions of a protocol, and outputs of some parties are used by other parties as inputs. This is where the doubts

about KEA in [21] arise. We suggest that KEA may still hold in such settings. We formalize the idea as an auxiliary-input KEA as follows.

Definition 3.1. *The* auxiliary-input KEA (AI-KEA) *is as follows. Let g be a generator of a prime order group $\langle g \rangle$. The value $k = \log |\langle g \rangle|$ is the security parameter. Let $\boldsymbol{A}_1, \cdots, \boldsymbol{A}_l$ be a group of polynomial-time algorithms that take as input g and g^a, where a is an unknown randomly chosen value. The value l is polynomial in k, and the number of output of each $\boldsymbol{A}_i$ is also polynomial in k. Any $\boldsymbol{A}_i, 1 \leq i \leq l$, can also take the output of any $\boldsymbol{A}_j, 1 \leq j \leq l$, as auxiliary input. Then there exists a polynomial-time "extractor" $\boldsymbol{E}$ that satisfies the following: when any $\boldsymbol{A}_i, 1 \leq i \leq l$, produces as output a pair of the form $(x, y), x \in \langle g \rangle$, $\boldsymbol{E}$ outputs the pair (x, y) along with an exponent r such that for sufficiently large k,*

$$\Pr\left[y = x^a \text{ and } g^r \neq x\right] < \frac{1}{Q(k)}$$

for any polynomial $Q()$.

Note that in the AI-KEA, the extractor $\mathbf{E}$ may need to access some other parties in $\{A_i : 1 \leq i \leq l\}$ besides the one which directly outputs (x, x^a). Roughly speaking, although no single party in $\{A_i : 1 \leq i \leq l\}$ "knows" r, all these parties together "know" r.

To further justify AI-KEA, we consider the following question: given multiple pairs of $(g_1, {g_1}^a)$, $(g_2, {g_2}^a), \cdots$, if one party outputs a pair (x, x^a), then what can the extractor extract from *this specific party*? With the similar idea underlying KEA, we assume that the only way to construct such a pair is to choose $r_1, r_2 \cdots$, and compute $x = {g_1}^{r_1} {g_2}^{r_2} \cdots$ and $y = {g_1}^{ar_1} {g_2}^{ar_2} \cdots$. We formalize the idea as the Generalized Knowledge-of-Exponent Assumption (GKEA).

Definition 3.2. *The* Generalized Knowledge-of-Exponent Assumption *is as follows: Let g be a generator of a prime order group $\langle g \rangle$. The value $k = \log |\langle g \rangle|$ is the security parameter. Given $T_n = \{(x_i, {x_i}^a) : 0 \leq i \leq n\}$, for any PPT algorithm $\boldsymbol{A}$, if $\boldsymbol{A}(T_n)$ outputs a pair (x, y), then there exists a PPT extractor $\boldsymbol{E}(T_n)$ which outputs $(x, y, c_0, \cdots, c_n)$, such that for any polynomial Q, it holds that*

$$\Pr\left[y = x^a \text{ and } \prod_{i=0}^{n} {x_i}^{c_i} \neq x\right] < 1/Q(k)$$

for k large enough.

Then we have the following result.

Theorem 3.3. *If GKEA holds, then the AI-KEA holds.*

Proof. (Sketch) Suppose in the AI-KEA setting that an $\mathbf{A}$ outputs (x, x^a) based on input pairs $(x_i, {x_i}^a)$. By GKEA, $\mathbf{E}$ can extract r_i such that $x = \prod {x_i}^{r_i}$. Then $\mathbf{E}$ does this recursively for each auxiliary input pair $(x_i, {x_i}^a)$. Note the relations between the auxiliary input pairs can be represented by a directed acyclic graph (DAG) where each pair is a node, and there is an edge from pair a to pair b if b is generated based on a. There is one "source" node (g, g^a) in the DAG. The recursive tracing process of $\mathbf{E}$ will end up with the node (g, g^a). Then $\mathbf{E}$ can compute r such that $x = g^r$ from the r_i values extracted during the recursive tracing process. Since there are a polynomial number of nodes in the DAG, $\mathbf{E}$ can finish the process in polynomial time. □

We note that there are cases where KEA is not applicable. For example, in ElGamal encryption scheme, the decryption algorithm will output pairs of the form (x, x^a) under a chosen ciphertext attack. In this case, if an adversary outputs a pair of the form (x, x^a), it is no longer reasonable to assume the adversary knows r such that $x = g^r$. The key point is that, when the adversary observes pairs of $(x_i, {x_i}^a)$ from previous sessions of the decryption of ElGamal scheme, these pairs are generated by the decryption oracle. However, in the simulation paradigm, the simulator has no access to the decryption oracle. In other protocols, e.g., DEG, such pairs (except for (g, g^a)) are generated by users to which a simulator has access in the simulation paradigm. Therefore, we need to distinguish the two types of pairs. Those generated by a party to which a simulator has no access in the simulation paradigm are called *non-extractable pairs*, and those generated by a party to which a simulator has access in the simulation paradigm are called *extractable pairs*. We conclude that KEA and AI-KEA are applicable for protocols where there is only one non-extractable pair (i.e., the initial (g, g^a) pair).

4 Conclusion

In this paper, we propose an extremely simple identification protocol and prove its security using KEA. Also, we clarified an issue about using KEA for protocols where an adversary can get auxiliary input. It was not clear if KEA is applicable to such cases. However, we gave an affirmative answer by presenting two variants of KEA, Generalized KEA (GKEA) and Auxiliary-Input KEA (AI-KEA). These variants of KEA clarify the proper use of KEA.

References

[1] M. Bellare, M. Fischlin, S. Goldwasser, and S. Micali. Identification protocols secure against reset attacks. Unpublished manuscript, available from `http://www-cse.ucsd.edu/users/mihir/papers/id-reset.html` [this is the full version of [2].].

[2] M. Bellare, M. Fischlin, S. Goldwasser, and S. Micali. Identification protocols secure against reset attacks. In *EUROCRYPT 2001 Proceedings*, volume 2045 of *Lecture Notes in Computer Science*, pages 495–511, Berlin, Heidelberg, New York, 2001. Springer.

[3] M. Bellare and A. Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 273–289. Springer, 2004.

[4] M. Bellare and A. Palacio. Towards plaintext-aware public-key encryption without random oracles. In Pil Joong Lee, editor, *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*, pages 48–62. Springer, 2004.

[5] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *EUROCRYPT 2000 Proceedings*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155, Berlin, Heidelberg, New York, 2000. Springer.

[6] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *CRYPTO '93 Proceedings*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249, Berlin, Heidelberg, New York, 1994. Springer.

[7] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.*, 33(1):167–226, 2004.

[8] I. Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In *CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, pages 445–456, London, UK, 1992. Springer-Verlag.

[9] A.W. Dent. The Cramer-Shoup encryption scheme is plaintext aware in the standard model. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 289–307. Springer, 2006.

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

[11] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In *CRYPTO '86 Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Berlin, Heidelberg, New York, 1987. Springer.

[12] L. Guillou and J.-J. Quisquater. A "paradoxical" identity-based signature scheme resulting from zero-knowledge. In *CRYPTO '88 Proceedings*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231, Berlin, Heidelberg, New York, 1990. Springer.

[13] S. Hada and T. Tanaka. On the existence of 3-round zero-knowledge protocols. In *CRYPTO '98: Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology*, pages 408–423, London, UK, 1998. Springer-Verlag.

[14] H. Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 546–566. Springer, 2005.

[15] T. Okamoto. Provably secure and practical identitication schemes and corresponding signature schemes. In *CRYPTO '92 Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53, Berlin, Heidelberg, New York, 1993. Springer.

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

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

[18] D.R. Stinson. Some observations on the theory of cryptographic hash functions. *Des. Codes Cryptography*, 38(2):259–277, 2006.

[19] D.R. Stinson and J. Wu. An efficient and secure two-flow zero-knowledge identification protocol. *Journal of Mathematical Cryptology*, 1:201–220, 2007.

[20] D.R. Stinson and J. Wu. A zero-knowledge identification and key agreement protocol. Cryptology ePrint Archive, Report 2007/116, 2007. `http://eprint.iacr.org/`.

[21] A.C.C. Yao, F.F. Yao, Y. Zhao, and B. Zhu. Deniable internet key-exchange. Cryptology ePrint Archive, Report 2007/191, 2007.