Zero-Knowledge Blind Identification For Smart Cards Using Bilinear Pairings

Amitabh Saxena and Ben Soh
Computer Science and Computer Engineering
La Trobe University
Bundoora, VIC, Australia 3086

Serguey Priymak
Applied Science Department
RMIT University
Melbourne, VIC, Australia 3000

October 7, 2005

Abstract

Traditionally in an identification scheme, a passive adversary watching the communication gains information intended only for the verifier. For instance, from watching the transcript in the Fiat-Shamir zero knowledge identification scheme, an adversary also learns the outcome of the protocol (i.e. whether the identification succeeds or not). We introduce the concept of blind identification that eliminate this disadvantage while still ensuring zero knowledge. Informally in such a scheme, only the verifier and the prover know the outcome of an identification protocol while a passive adversary watching the entire communication does not gain any useful information. To achieve this, our proposed protocol differs from traditional identification schemes in two contrasting features: (1) We require the verifier to toss his coins *before* the prover. (2) The coin tosses of the verifier are secret while the coin tosses of the prover are not. As a natural extension of the single user identification, we present the concept of 'all or none' group identification protocol that can be used to authenticate together an arbitrary number of users in a batch such that if the identification fails, it is impossible for the users to know who cheated. Finally, we present some interesting applications including hidden signatures, anonymous seller credit card transactions, two-round authenticated key agreement, authenticated and deniable encryption and identity based cryptography.

Our protocol is secure assuming the hardness of a problem closely related to the Diffie-Hellman Problem in bilinear groups. The cryptographic primitives can be efficiently encapsulated in smart cards designed for Elliptic Curve Cryptography (ECC). The private key must be included in a tamperproof device inside the smart card.

1 Introduction

The user of zero-knowledge proofs for identification has interesting implications. For instance, a verifier cannot later impersonate the prover using the transcript of the proof.

Zero-knowledge identification does not solve all the problems since it is still susceptible to a man-in-the-middle attack, where a passive adversary simply relays the messages between the prover and the verifier. In a typical scenario, this adversary has the same information as the verifier. We present a slightly stronger variation of zero-knowledge identification where a passive adversary does not learn anything new from observing the transcript of the protocol. This paper has two motivations; firstly, to provide a secure 'pairing' variant of identification schemes based on the discrete logarithm problem and secondly, to provide an efficient smart card based zero-knowledge identification scheme which does not leak any information to a passive adversary. In other words, the scheme allows users to correctly identify a prover to a verifier such that: (a) If both the prover and verifier are honest, the protocol leaks only the *same* one bit of information (that the prover's statement is indeed true) after arbitrarily many runs and (b) The protocol leaks this one bit of information *only* to an honest verifier while a passive adversary watching the entire communication still gains no knowledge about the correctness of prover's statement after arbitrarily many runs. To do this, we require the verifier to toss secret coins while requiring the prover to toss public coins. Additionally, we require that the verifier toss his coins before the prover. We also give some extensions of the scheme including two-way authentication, key agreement, single-key signcryption (which allows a user Alice, for example to send a signed and encrypted message to another user Bob without involving Bob's keys at all!), hidden signatures (which allow Alice to send a message along with a hidden signature to Bob) and online credit card/cheque payments.

The cryptographic primitives can be implemented efficiently in smart cards; one-way identification requires only two elliptic curve point multiplications while two-way identification requires three multiplications and two pairing computations for each user. Unlike most pairing based schemes which involve identity based cryptography, our model is based on a standard certificate based infrastructure. Due to this we lose certain benefits offered only by identity based mechanisms (like implicit key authentication). However, at the same time we are unaffected by the key escrow problem inherent in all identity based systems.

The rest of the paper is organized as follows. In section 2 we give some notations and concepts necessary to describe our protocol. In section 3 we present the cryptographic primitives used in our protocol along with the necessary hardness assumptions. We describe the underlying Public Key Infrastructure required for our protocol in section 4. Finally, we present our scheme in section 5 and provide several extensions in sections 7 and 9.

2 Background

In this section, we present a simple identification scheme using a public key cryptosystem. Assume that Alice and Bob are two users and Alice wishes to identify herself to Bob. We only consider one-way identification and ignore the case of Bob identifying himself to Alice.

First we give some notation. If $\mathbb{A}$ is a non-empty set, then $x \leftarrow \mathbb{A}$ denotes that x has been uniformly chosen in $\mathbb{A}$. A *round* of a protocol involves the exchange of one message. A sequence of two synchronous (ordered) message transmissions constitutes two separate rounds while any number of asynchronous messages (i.e. messages that can be unordered) are part of the same round. A single message passing is a one-round protocol.

2.1 Basic Two-Round Identification

Alice has public encryption function E_a and a secret decryption function D_a.

1. Alice begins by claiming to know D_a.
2. Bob generates a random challenge message m and encrypts it using E_a to get ciphertext $c = E_a(m)$. Bob sends c to Alice over an insecure channel.
3. Alice computes $m' = D_a(c)$ and sends back m' to Bob over the same insecure channel.
4. Bob accepts if $m = m'$.

This protocol is correct and sound but has an inherent disadvantage. It may be a security requirement of Alice that only the sender of c (whether it is Bob or not) may know her identity. An adversary watching the communication can also check that $E_a(m') = c$ and consequently obtain information about the outcome of the identification that only Bob was supposed to have. The Fiat-Shamir identification protocol [1] also suffers from this drawback. Additionally in this protocol, an adversary is able to obtain decryption of an arbitrary ciphertext intended for Alice. A modified version of the protocol overcomes this problem by using a hash function $\mathcal{H} : \{0,1\}^* \mapsto \{0,1\}^*$. In step 3, instead of sending m, Alice sends $m'' = \mathcal{H}(m')$ to Bob who accepts if $m'' = \mathcal{H}(m)$. However, this protocol still does not satisfy our strong notion of zero-knowledge because Alice's reply is deterministic. If an adversary replays the same message c to Alice later on and receives the same m'', he can be reasonably confident that he is talking to Alice without knowing m. To avoid this attack, Alice must insert some randomness into her reply each time. In the second protocol described below, we incorporate this.

2.2 Modified Two-Round Identification

Alice has public encryption function E_a and a secret decryption function D_a. Let $\mathcal{H} : \{0,1\}^* \mapsto \{0,1\}^*$ be a cryptographic hash function.

1. Alice begins by claiming to know D_a.
2. Bob generates a random challenge message m and encrypts it using E_a to get ciphertext $c = E_a(m)$. Bob sends c to Alice over an insecure channel.
3. Alice generates a random $r \leftarrow \{0,1\}^*$ and computes $s = \mathcal{H}(D_a(c)\|r)$. She sends back the tuple $\langle r, s\rangle$ to Bob over the same insecure channel.
4. Bob accepts if $s = \mathcal{H}(m\|r)$.

The protocol is unconditionally secure if $\mathcal{H}$ is a random oracle. Observe that Alice must toss coins to insert randomness r into her reply to avoid the known ciphertext attack. The identification protocol presented in this paper is based on a similar idea. However, instead of the difficulty of inverting the public encryption function, we rely on the difficulty of computing discrete logarithms in certain groups. Like the above protocol, our scheme is two-round and begins by the verifier tossing secret coins while the prover then tosses public coins.

Identification schemes based on the discrete logarithm problem have been proposed earlier, for example in [2, 3]. All these schemes, however, assume that the underlying DDH problem is computationally hard. Due to this, some of their security properties are lost when used in a pairing based scenario where the DDH is easy. Moreover, in most of the schemes, a passive adversary knows everything that the provers and verifiers know. It may be a security requirement of Alice and Bob that a passive adversary must not learn the outcome of the identification. The purpose of this paper is to present an identification protocol for smart cards using bilinear pairings satisfying this requirement and not relying on hash functions or random oracles. We coin the term *blind identification* to denote a protocol where a passive adversary cannot learn the outcome of the protocol.

3 Bilinear Pairings

Pairing based cryptography is based on the existence of efficiently computable non-degenerate bilinear maps which can be abstractly described as follows: Let $\mathbb{G}_1$ be a cyclic additive group of prime order q and $\mathbb{G}_2$ be a cyclic multiplicative group of the same order. Assume that computing the discrete logarithm in both $\mathbb{G}_1$ and $\mathbb{G}_2$ is hard. A bilinear pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$ that satisfies the following properties [4, 5]:

1. *Bilinearity*: $e(aP, bQ) = e(P,Q)^{ab}\ \forall P, Q \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q$
2. *Non-degeneracy*: $P \neq 0 \Rightarrow e(P,P) \neq 1$
3. *Computability*: e is efficiently computable

The above properties also imply:

$$e(P+Q, R) = e(P,R) \cdot e(Q,R)\ \forall P, Q, R \in \mathbb{G}_1$$

$$e(P, Q+R) = e(P,Q) \cdot e(P,R)\ \forall P, Q, R \in \mathbb{G}_1$$

Typically, the map e will be derived from either the Weil or Tate pairing on an elliptic curve over a finite field. Despite the complex mathematics involved in constructing such maps, cryptographic protocols based on pairings can be described entirely without ever referring to the actual implementation. We refer the reader to [4, 5, 6] for more details. Pairings and other parameters should be selected in proactive for efficiency and security. For appropriately selected parameters, we will assume the following assumptions hold unconditionally.

3.1 Assumptions

1. *Discrete Logarithm Assumption*: The *Discrete Logarithm Problem (DLP)* in $\mathbb{G}_1$ (and consequently $\mathbb{G}_2$) is intractable. In other words, given any two elements $P, Y \in \mathbb{G}_1$, computing $x \in \mathbb{Z}_q^*$ such that $Y = xP$ is hard.

2. *Diffie-Hellman Assumption*: Given $P, Y, R \in \mathbb{G}_1$ such that $Y = xP$ and $R = rY$ for unknowns $x, r \in \mathbb{Z}_q^*$, computing $U = rP$ is infeasible. This is the *Diffie-Hellman Problem (DHP).*

3. *Extended Diffie-Hellman Assumption*: Given $P, Y, R \in \mathbb{G}_1$ such that $Y = xP$ and $R = rY$ for unknowns $x, r \in \mathbb{Z}_q^*$, computing $U = r^2P$ is infeasible. We call this the *Extended Diffie-Hellman Problem (EDHP).*

4. *Extended Decisional Diffie-Hellman Assumption*: Given $P, Y, R, U \in \mathbb{G}_1$ such that $Y = xP$ and $R = rY$ for unknowns $x, r \in \mathbb{Z}_q^*$, deciding if $U = r^2P$ with probability $> 1/2$ is infeasible. This is the *Extended Decisional Diffie-Hellman Problem (EDDHP*). (We note that the Decisional Diffie-Hellman Problem (DDHP), which requires deciding if $U = rP$ is easy in this case, since it only requires two pairing computations to decide if $e(P, R) = e(Y, U)$. All pairing based schemes make use of this observation [4]).

5. *Linear Diffie-Hellman Assumption*: Given $P, Y, R \in \mathbb{G}_1$ such that $Y = xP$ and $R = rY$ for unknowns $x, r \in \mathbb{Z}_q^*$, computing *any* pair $\langle Z, Q\rangle$ such that $Z = rP + xQ$ is infeasible. We call this the *Linear Diffie-Hellman Problem (LDHP).*

6. *Linear Decisional Diffie-Hellman Assumption*: This relates to the decisional variant of the LDHP. Given $P, Y, R, Z, Q \in \mathbb{G}_1$ such that $Y = xP$ and $R = rY$ for unknowns $x, y \in \mathbb{Z}_q^*$, deciding if $Q = yP + xR$ with probability $> 1/2$ is infeasible. We call this the *Linear Decisional Diffie-Hellman Problem (LDDHP).*

3.2 Problem Hierarchy

Before describing our identification protocol, we briefly describe the observed relationship between the different problems considered above. An arrow indicates a reduction with the tail end representing the harder problem (note that some authors use the opposite notation where the tail indicates the easier problem). For each of the relationships, we consider P to be fixed.

$$\begin{tikzcd}
& & EDHP \arrow[r, Rightarrow] \arrow[dl, Rightarrow, shift left] \arrow[dd, Rightarrow] & EDDHP \\
DLP \arrow[r, Rightarrow] & DHP \arrow[ur, Rightarrow, shift left] \arrow[dr, Rightarrow] & & \\
& & LDHP \arrow[r, Rightarrow] & LDDHP
\end{tikzcd}$$

To see that DHP $\Rightarrow$ EDHP, let $\text{DHP}_P(Y, R)$ be the output of an oracle that solves the DHP for some fixed P and arbitrary Y, R. We can use this oracle to construct another oracle $\text{EDHP}_P(Y, R)$ that solves the EDHP for the same tuple $\langle P, Y, R\rangle$ as follows: $\text{EDHP}_P(Y, R) = \text{DHP}_P(\text{DHP}_P(\text{DHP}_P(Y, R), Y), R)$.

To see that EDHP $\Rightarrow$ DHP, let $\text{EDHP}_P(Y, R)$ be the output of an oracle that solves the EDHP for some fixed P and arbitrary Y, R. We can use this oracle to construct a $\text{DHP}_P(Y, R)$ oracle that solves the DHP for the same tuple $\langle P, Y, R\rangle$ as follows: $\text{DHP}_P(Y, R) = \frac{1}{2}(\text{EDHP}_P(Y, Y+R) - \text{EDHP}_P(Y, R) - P)$.

Finally, we note the DHP is considerably harder than the LDHP since there does not appear to be any simple oracle-based reduction from the LDHP to the DHP. In fact the results of Boneh et al. [7] indicate that making a $\text{DHP}_P(Y,R)$ oracle from an $\text{LDHP}_P(Y,R)$ oracle is as hard as solving the DHP itself.[1]

4 Setup PKI

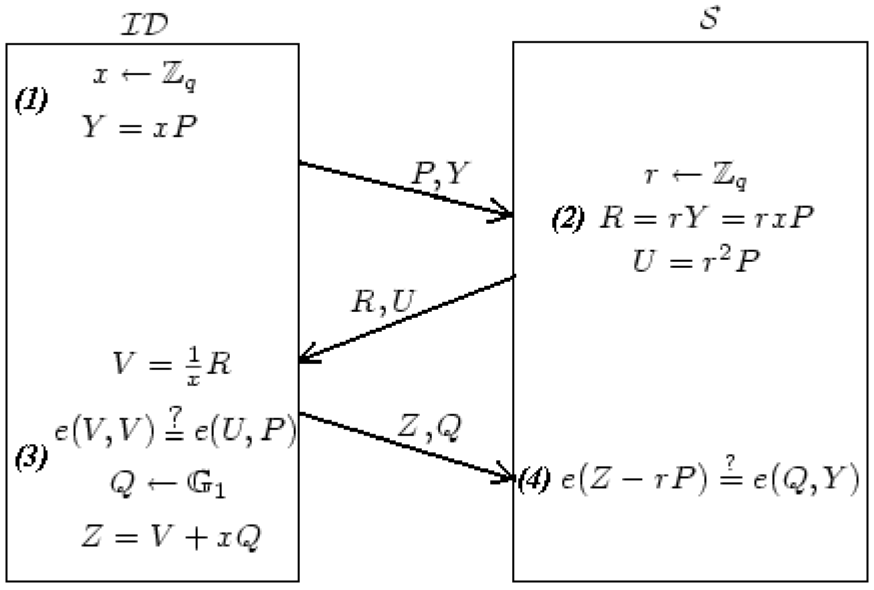
In the rest of the discussion, we will using a PKI which will be setup as follows: A central authority is responsible for generating the security parameters. A trusted CA is responsible for certifying the public keys. To participate in the protocol each user must have a certified public key (the process of certification is outside the scope of our protocol). The setup proceeds as follows:

1. Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$ be a bilinear mapping and $P \in \mathbb{G}_1$ be a generator of $\mathbb{G}_1$. The parameters $\langle e, \mathbb{G}_1, P\rangle$ are generated by the trusted authority and made public in an authentic way.

2. Each participant $\mathcal{ID}_i$ generates $x_i \leftarrow \mathbb{Z}_q$ as the private key. The corresponding public key is $Y_i = x_iP \in \mathbb{G}_1$. Each user also obtains a certificate from the CA linking the identity $\mathcal{ID}_i$ and the public key Y_i. In other words, the CA fixes the pairs $\langle Y_i, \mathcal{ID}_i\rangle$.

5 Blind Identification

Assume that user $\mathcal{ID}$ having secret key $x \in \mathbb{Z}_q$ and public key $Y = xP \in \mathbb{G}_1$ wants to prove to server $\mathcal{S}$, the knowledge of x. Additionally, $\mathcal{ID}$ wants to ensure that no one except the verifier $\mathcal{S}$ gets convinced of this fact from watching the communication (in other words, the proof needs to be verifier dependent and a dishonest verifier does not get convinced about the statement). We will assume the infrastructure of section 4. The protocol is graphically described in figure 1.



$\mathcal{ID}$
$\mathcal{S}$
(1)
$x \leftarrow \mathbb{Z}_q$
$Y = xP$
P, Y
$r \leftarrow \mathbb{Z}_q$
(2) $R = rY = rxP$
$U = r^2P$
R, U
$V = \frac{1}{x}R$
(3)
$e(V,V) \stackrel{?}{=} e(U,P)$
$Q \leftarrow \mathbb{G}_1$
$Z = V + xQ$
Z, Q
(4) $e(Z - rP) \stackrel{?}{=} e(Q,Y)$

Figure 1: Blind Identification

[1]The task of a $\text{DHP}_P(Y,R)$ oracle is to compute rP while the task of an $\text{LDHP}_P(Y,R)$ oracle is to compute many (say n) pairs (Z_i, Q_i) such that $Z_i = rP + xQ_i$ $(1 \le i \le n)$. Thus to solve the DHP using the LDHP oracle would be equivalent to extracting rP from $rP + xQ_i$ given Q_i $(1 \le i \le n)$. The aggregate signature scheme of [7] assumes that such an extraction is impossible if the value of n is polynomially bounded.

1. $\mathcal{ID}$ starts by claiming to know $x \in \mathbb{Z}_q$, the discrete logarithm of $Y \in \mathbb{G}_1$ to base P.
2. The verifier $\mathcal{S}$ generates $r \leftarrow \mathbb{Z}_q$ and computes $R = rY$ and $U = r^2P$. It sends $\langle R, U\rangle$ as its challenge to $\mathcal{ID}$. Typically the challenge should have a very short lifetime.
3. $\mathcal{ID}$ computes $V = \frac{1}{x}R$ and verifies $e(V, V) = e(U, P)$. If this test passes, $\mathcal{ID}$ is ensured that R was indeed randomly generated. $\mathcal{ID}$ generates $Q \leftarrow \mathbb{G}_1$ and computes $Z = V + xQ$. It sends $\langle Z, Q\rangle$ as its proof to $\mathcal{S}$.
4. $\mathcal{S}$ accepts if $e(Z - rP, P) = e(Q, Y)$.

6 Security Proof (Sketch)

We claim that this test will pass with a high probability if and only if $\mathcal{ID}$ knows x *and* $\mathcal{S}$ knows r. If the former is false, the verification fails with a high probability while if the latter is false, no knowledge about x is given out in the process and the outcome of the proof is undecidable.

1. Correctness: The properties of bilinear maps ensure that the verification is always successful if no one cheats.
2. Soundness: We will analyze the soundness assuming that one of the parties is dishonest. The trivial case when both parties are dishonest is ignored. The soundness property holds because:
   (a) Dishonest Verifier: A dishonest verifier will generate R non-randomly. In other words, a dishonest verifier will not know r corresponding to R. Due to this it will be hard to generate U such that $e(\frac{1}{x}R, \frac{1}{x}R) = e(U, P)$. Thus a dishonest verifier will not be able to make $\mathcal{ID}$ accept the challenge as valid (see appendix A.2 for an informal proof).
   (b) Dishonest Prover: Given P, xP, rxP, Computing a pair Z', Q' such that $Z' = rP + xQ'$ is infeasible without knowledge of r or x due to the hardness of the LDHP as shown in theorem 4.4 of [8] (cf. aggregate extraction). Additionally, an honest verifier keeps r secret. Thus, the proof is sound from a verifier's view (see appendix A.1 for an informal proof).
3. Zero Knowledge: We will analyze the zero knowledge property of the protocol from the following perspectives. The case of dishonest verifier is ignored due to the reasoning above (item 2a).
   (a) Honest Verifier Zero-Knowledge: An honest verifier can generate an accepting transcript $\{Z', r, Q', R, U, Y, P\}$ without interaction with a prover as follows: Let $r, \alpha \leftarrow \mathbb{Z}_q$, $R = rY$, $U = r^2P$, $Q' = \alpha P$ and $Z' = rP + \alpha Y$. It is easy to see that the simulated and real distributions are indistinguishable. Also observe that an unlimited number of accepting transcripts can be generated using the same coin tosses r.
   (b) Honest Verifier Secrecy: As shown in appendix B, it is impossible for a passive adversary to decide the honesty of the verifier. That is, given P, xP, R, U, deciding if $e(\frac{1}{x}R, \frac{1}{x}R) = e(U, P)$ is infeasible without knowledge of r or x due to the hardness of the DHP in $\mathbb{G}_1$.
   (c) Honest Prover Secrecy: It is not possible for a passive adversary to decide the honesty of the prover: Given P, xP, rxP, r^2P, Q, Z, deciding if $Z = rP + xQ$ is infeasible without knowledge of r or x due to the hardness of the LDDHP in $\mathbb{G}_1$ as shown in appendix B.

7 Blind Group Identification

This scheme enables a group of users to identify themselves to a server such that: (a) The identification test passes if none of the users cheat, (b) if any users cheat, the test will fail with a high probability, (c) it is not possible for the server or the users to know which person cheated. An important application for this type of scheme is in the following type of group systems: Assume that two users Alice and Bob want to identity themselves jointly to a server (for example, because they don't trust each other to individually login to the server without the other's approval). Alice wants to ensure that the identification succeeds if and only if the other user is really Bob. Bob has a similar requirement.

Assume that $\{\mathcal{ID}_1, \mathcal{ID}_2, \ldots \mathcal{ID}_n\}$ are the set of users who want to jointly identify themselves. It is necessary that each user $\mathcal{ID}_i$ must have a certified public key Y_i as described earlier. The goal of the protocol is that all users will simultaneously identify themselves to a server $\mathcal{S}$.

In other words, each user $\mathcal{ID}_i$ will prove possession of the discrete logarithm x_i of Y_i (to base P) such that $\mathcal{S}$ cannot be convinced about any of the individual statements separately. That is, the proof is valid only on all the statements together: "$\mathcal{ID}_i$ knows x_i" $\forall i : 1 \leq i \leq n$ but not on any of the individual statements like "$\mathcal{ID}_1$ knows x_1" or "$\mathcal{ID}_2$ knows x_2" independently of the others (we intuitively coin the term *additive zero knowledge* for this [9]). We will assume the infrastructure of section 4. The identification is done as follows:

1. The n provers $\mathcal{ID}_1, \mathcal{ID}_2 \ldots \mathcal{ID}_n$ start by claiming to $\mathcal{S}$ that they know the discrete logarithms $x_1, x_2, \ldots x_n \in \mathbb{Z}_q$ of $Y_1, Y_2, \ldots Y_n \in \mathbb{G}_1$ (to base P) respectively.

2. The verifier $\mathcal{S}$ generates $r_1, r_2, \ldots r_n \leftarrow \mathbb{Z}_q$ and computes $R_i = r_i Y_i$ and $U_i = r_i^2 P$. It makes the list of challenges $\langle \mathcal{ID}_i, R_i, U_i \rangle$ public.

3. Each $\mathcal{ID}_i$ computes $V_i = \frac{1}{x_i} R_i$ and checks that $e(V_i, V_i) = e(U_i, P)$. If this test passes, it generates $Q_i \leftarrow \mathbb{G}_1$ and computes $Z_i = V_i + x_i Q_i$

4. All users then collaborate to jointly compute the value $Z = \sum_{j=1}^{j=n} Z_j$. This computation is hidden from $\mathcal{S}$ so that individual values Z_j are effectively hidden the it's view. The combined proof $\{Z, Q_1, Q_2 \ldots Q_n\}$ is sent to $\mathcal{S}$.

5. $\mathcal{S}$ accepts if $e(Z - \sum_{j=1}^{j=n} r_j P, P) = \prod_{j=1}^{j=n} e(Q_j, Y_j)$.

8 Security Proof (Sketch)

We claim that this test will pass if and only if each $\mathcal{ID}_i$ knows x_i. To summarize the goals of the protocol, the individual users can jointly authenticate themselves to the server such that:

(a) If all users are honest, the server always accepts.

(b) If any of the users are dishonest, the server rejects with a high probability.

(c) The protocol is zero knowledge. It is not possible for anyone (including the server) to know which user cheated.

(d) Collusions are possible between users but not with the server (the server is trusted).

The protocol is secure based on the following observations:

1. Correctness: The properties of bilinear maps ensure that the verification is always successful if none of $\mathcal{ID}_i$ cheat.

2. Soundness: We discuss the soundness using a similar reasoning as above. The soundness property holds because:

   (a) Dishonest Prover: Computing individual proofs $\langle Z_i, Q_i \rangle$ without x_i or r_i is infeasible using similar reasoning for the single user scenario (section 5). Using the idea of aggregate signatures of [8], the same applies to the multiuser case. Consequently the verifier will reject.

(b) Dishonest Verifier: A dishonest verifier will generate R non-randomly and will therefore not know r_i corresponding to R_i. Due to this it will be hard for this verifier to generate U_i such that $e(\frac{1}{x_i}R_i, \frac{1}{x_i}R_i) = e(U_i, P)$ due to the hardness of the EDHP. Thus a dishonest verifier will not be able to make anyone accept his/her challenge as valid. The above reasoning for single user identification can be extended here.

3. Zero Knowledge: As before, we will analyze the zero knowledge property from the following perspectives:

   (a) Honest Verifier Zero Knowledge: $\mathcal{S}$ can generate a valid accepting transcript on its own corresponding to $\{r_i, R_i, U_i\}$ $\forall i : 1 \leq i \leq n$ as follows: $\mathcal{S}$ generates $\alpha_i \leftarrow \mathbb{Z}_q$ $\forall i$ and computes $Q_i = \alpha_i P$, $R_i = r_i Y_i$. Then $Z = \sum_{j=1}^{j=n} r_i P + \alpha_i Y_i$.

   (b) Honest Verifier Secrecy: We require that it is impossible for a passive adversary to decide the honesty of the verifier. The reasoning for the single user case can be extended here. That is, given $P, x_i P, R_i, U_i$, deciding if $e(\frac{1}{x_i}R_i, \frac{1}{x_i}R_i) = e(U_i, P)$ is infeasible without knowledge of r_i or x_i due to the hardness of the DHP in $\mathbb{G}_1$ [7].

   (c) Honest Prover Secrecy: Assume that all the provers are honest and thus, $\mathcal{S}$ will eventually accept. We require that it is impossible for a passive adversary (including the provers) to decide the honesty some prover. We note that given $P, x_i P, r_i x_i P, r_i^2 P, Q_i, Z_i$, deciding if $Z_i = r_i P + x_i Q_i$ is infeasible without knowledge of r or x due to the hardness of the LDDHP in $\mathbb{G}_1$ as shown in appendix B. Thus a passive adversary cannot decide the outcome of the identification.

   (d) Dishonest Prover Secrecy: Assume that some of the provers are dishonest. In this case, deciding if any given $Z_i = \frac{1}{x_i}R_i + x_i Q_i$ is infeasible without knowledge of x_i or r_i due to the decisional linear Diffie-Hellman assumption. That is, given $P, x_i P, r_i x_i P, Q, Z_i \in \mathbb{G}_1$, deciding if $Z_i = r_i P + x_i Q$ is infeasible without knowing at least one of $\{x_i, r_i\}$. Therefore if $\mathcal{S}$ rejects, none of the provers know which pairs $\langle Z_i, Q_i \rangle$ correspond to invalid proofs (if the individual coin tosses r_i of $\mathcal{S}$ are kept secret and $\mathcal{S}$ is honest, no information is leaked to the provers). Similarly if the individual values Z_i are kept secret (from $\mathcal{S}$), the identity of the dishonest provers is still concealed since computing individual proofs Z_i just from $Z, Q_1, Q_2 \ldots Q_n$ such that $Z_i = r_i P + x_i Q_i$ $\forall i$ is infeasible without knowledge of each x_i due to the hardness of the DHP in $\mathbb{G}_1$ as shown in theorem 4.4 of [8] (cf. aggregate extraction). Consequently, even the verifier $\mathcal{S}$ does not have the ability to decide which of the provers are dishonest.

   Finally, if the joint computation of Z is carried out in a way that any one individual prover or a small coalition of provers can know Z_i's for only a small fraction of users, the identities of dishonest provers can still be effectively hidden, even if $\mathcal{S}$ can be coerced to reveal all the coin tosses r_i.

9 Other Extensions

In this section we will provide several extensions of our scheme. We refer to the definitions of sections 4 and 5. The private keys x_i can either be generated by the users or by a trusted authority. The public key Y_i are assumed to be certified in the former case. Note that the identification is a two-round protocol, with the verifier sending the challenge R in the first step and the prover sending the response Z, Q in the second step. The private key for each smart card is encapsulated in a tamperproof chip. Signing access to this key is given via some access control mechanism like a PIN number. The corresponding public key is also present in the smart card along with a certificate. Smart cards may be purchased from a (reputed) third party and must be registered with the relevant authority (like a bank) before they can be used. To register a smart card, the authority simply provides a certificate.

9.1 Authenticated Encryption (Signcryption)

Assume that user $\mathcal{ID}$ is identifying itself to the server $\mathcal{S}$ using the blind identification scheme of section 5. $\mathcal{ID}$ can encrypt a random message $M \in \mathbb{G}_1$ intended for $\mathcal{S}$ using the challenge R as follows: Let $M = xQ$. Using this relation, $\mathcal{ID}$ computes $Q = \frac{1}{x}M$. It then checks that (R, U) is indeed a valid challenge by computing $V = \frac{1}{x}R$ and checking that $e(V, V) = e(U, P)$. If this check passes, it computes $Z = V + xQ$ as before. The authenticated ciphertext is (Z, Q). An honest verifier $\mathcal{S}$ can compute the message M as follows: First it checks that the identification condition is true (i.e. it checks that (Z, Q) is indeed an accepting configuration). Then it computes $M = Z - rP$. Authentication is provided due to the zero-knowledge property; the verifier is assured that the sender is indeed $\mathcal{ID}$. It may appear that non-repudiation as such is not provided because $\mathcal{S}$ cannot later prove in a court that the message was sent by $\mathcal{ID}$ since $\mathcal{S}$ could very well have generated an accepting transcript without interaction with the prover by simulating the entire protocol.

However, note that in most cases when the transcript is simulated, the resulting message M will be meaningless. If $\mathcal{S}$ presents the tuple $\langle M, Q\rangle$ in a court, a judge can verify that that $e(M, P) = e(Q, Y)$. If the message is meaningful and the equality is valid, the judge is convinced that Q was indeed computed by $\mathcal{ID}$ and not by $\mathcal{S}$. Hence Q also serves as a signature on M.[2] The signature and encryption scheme is secure against Chosen Plaintext Attacks (IND-CPA) under the Diffie-Hellman assumption [4]. However, it is insecure against Chosen Ciphertext Attacks (IND-CCA) because (a) existential forgery of signatures is always possible and (b) given two ciphertext $\{(Z, Q), (Z', Q')\}$ and one of the plaintexts $\{xQ, xQ'\}$, it is always possible to associate it with its ciphertext.

Despite having the disadvantage of being IND-CCA insecure, our scheme offers an interesting feature; if in addition to $\langle M, Q\rangle$, the verifier $\mathcal{S}$ also produces $\langle Z, r\rangle$ in the court and Z has been signed by $\mathcal{ID}$ (using some scheme, which is irrelevant to us), then $\mathcal{S}$ can also claim that the message was *directly* received from $\mathcal{ID}$. To validate this, the judge checks if $e(Z - rP, P) = e(Q, Y)$. This tells the judge that the message was signed by $\mathcal{ID}$ and also *intended* for the holder of r (which turns out to be $\mathcal{S}$ in this case) since even $\mathcal{ID}$ does not have the ability to compute r. Observe that this encryption/non-repudiation is provided from using only $\mathcal{ID}$'s certified public key(s) in contrast to most other schemes that require a certified key for each party. This feature is a new type of non-repudiation (or commitment) that can be used in electronic payment systems as described later.

9.2 Signatures

To construct an IND-CCA secure signature scheme from the protocol of section 5, we simply remove the verifier from the protocol and set its challenge $R = 0 \in \mathbb{G}_1$. The prover's response is extracted from the message using a hash function $\mathcal{H} : \mathbb{G}_1 \mapsto \mathbb{G}_1$. That is, $Q = \mathcal{H}(M)$ where $M \in \mathbb{G}_1$ is a message. The signature of user $\mathcal{ID}$ is $S = xQ$. This is exactly the short signature scheme of Boneh et al. [4]. To verify that (M, S) is a valid message-signature pair, check that $e(S, P) = e(\mathcal{H}(M), Y)$.

9.3 Hidden Signatures

In the scenario where user $\mathcal{ID}$ identifies to the server $\mathcal{S}$ (section 5), $\mathcal{ID}$ can also send plaintext messages along with hidden signatures such that only $\mathcal{S}$ can extract the signature. Of course, once extracted, the signatures provide the same non-repudiation as ordinary signatures. Like the previous scheme, the message to be signed is $M \in \mathbb{G}_1$ and $Q = \mathcal{H}(M)$. However, in this case, the verifier's challenge is not ignored. As always, the prover first computes $V = \frac{1}{x}R$ and checks that $e(V, V) = e(U, P)$. The hidden signature of $\mathcal{ID}$ on M is then $Z = V + xQ$. The tuple $\langle M, Z\rangle$ is sent to $\mathcal{S}$ who can extract the original signature $S = Z - rP$. The verification condition is $e(S, P) = e(\mathcal{H}(M), Y)$ like before.

[2]Observe that even though the message M is encrypted, the signature $\frac{1}{x}M = Q$ is not. Suppose the security requirement is that even the signature must be encrypted. This is easily done by redefining $Q = M + rP$. Then we have $Z = rP + xQ = rP + xM + rxP$. In this case the recipient extracts the message $M = Q - rP$ and the signature $S = xM = Z - rP - R$. To verify a signature we check if $e(S, P) = e(M, Y)$

9.4 Authenticated Key Agreement

Using the protocol of section 5, authenticated key agreement between any two parties is possible. User $\mathcal{ID}_a$ having public key $x_a P_a$ and private key x_a wants to establish a shared key with user $\mathcal{ID}_b$ having public key $x_b P_b$ and private key x_b. The protocol is essentially an extension of the two-round identification protocol (it is possible that $P_a = P_b$). We provide two variants, the first for illustrative purposes.

9.4.1 Three-Round Key Agreement

This protocol requires three rounds (or three message exchanges) and is based on the traditional model for two-way authentication.

1. To initiate the protocol, $\mathcal{ID}_a$ generates $r_a \leftarrow \mathbb{Z}_q$ and computes $R_a = r_a Y_b$ and $U_a = r_a^2 P_b$. It sends $\langle R_a, U_a \rangle$ to $\mathcal{ID}_b$
2. $\mathcal{ID}_b$ computes $V_b = \frac{1}{x_b} R_a$ and verifies that $e(V_b, V_b) = e(U_a, P_b)$. If this test passes, $\mathcal{ID}_b$ generates $Q_b \leftarrow \mathbb{G}_1$, $r_b \leftarrow \mathbb{Z}_q$, computes $Z_b = V_b + x_b Q_b$ and $R_b = r_b Y_a$. It sends $\langle Z_b, Q_b, R_b \rangle$ to $\mathcal{ID}_a$.
3. $\mathcal{ID}_a$ checks that $e(Z_b - r_a P_b, P_b) = e(Q_b, Y_b)$. If this test passes, $\mathcal{ID}_a$ accepts $\mathcal{ID}_b$'s authentication, computes $Z_a = \frac{r_a}{x_a} R_b = r_a r_b P_a$ and sends Z_a to $\mathcal{ID}_b$.
4. $\mathcal{ID}_b$ accepts $\mathcal{ID}_a$'s authentication if $e(R_a, r_b P_a) = e(Z_a, Y_b)$. (In other words, $\mathcal{ID}_b$ checks that $\langle Z_a, Y_b, R_a, r_b P_a \rangle$ is indeed a valid DDH tuple). After this step, both parties are authenticated to each other. The shared key can be either $r_a P_b$ or $x_b Q_b$.

In the first three steps, $\mathcal{ID}_b$ identifies itself to $\mathcal{ID}_a$. In the fourth step $\mathcal{ID}_a$ identifies itself to $\mathcal{ID}_b$ by proving the knowledge of r_a, the discrete logarithm of R_a to base Y_b that was sent in step 1 (since a correct value Z_a simultaneously proves knowledge of x_a and r_a). An active adversary may still be able to substitute R_b sent in the second step with a new value R_b' without detection in the third step where $\mathcal{ID}_a$ accepts $\mathcal{ID}_b$'s authentication. However, this attack is useless. Firstly, observe that the adversary is unable to make $\mathcal{ID}_a$ use a chosen shared key for communication with $\mathcal{ID}_b$. The fourth step ensures that such an adversary cannot go undetected, since if this substitution attack was carried out, the adversary will not be able to send a correct value of Z_a' as its response to $\mathcal{ID}_b$ which is needed for two-way authentication. We note that the protocol is still zero-knowledge because a passive adversary is unable to decide if the authentication was successful after watching the communication. We also observe that it is possible to combine the first and last steps together as demonstrated in the next variant.

9.4.2 Two-Round Key Agreement

Using the protocol of section 5, a two-round authenticated key agreement is also possible. As before, user $\mathcal{ID}_a$ having public key $x_a P_a$ and private key x_a wants to establish a shared key with user $\mathcal{ID}_b$ having public key $x_b P_b$ and private key x_b. The protocol is essentially a proof of 'knowledge of knowledge' and is unconditionally secure under the following scenario: $\mathcal{ID}_b$ proves knowledge of $\mathcal{ID}_a$'s knowledge. That is, $\mathcal{ID}_a$ initiates by saying "I know x_a and r_a" and $\mathcal{ID}_b$ replies by saying "I know x_b if you know x_a and r_a". Let $\mathcal{H} : \mathbb{G}_1 \mapsto \mathbb{G}_1$ be a cryptographic hash function.

1. $\mathcal{ID}_a$ generates $r_a \leftarrow \mathbb{Z}_q$ and computes $R_a = r_a Y_b$, $U_a = r_a^2 P_b$ and $Z_a = r_a P_b + x_a \mathcal{H}(R_a)$. It initiates the protocol by sending $\langle R_a, U_a, Z_a \rangle$ to $\mathcal{ID}_b$. Essentially, $\mathcal{ID}_a$ simulates the identification protocol with itself and sends part of the transcript to $\mathcal{ID}_b$.
2. On receiving $\langle R_a, U_a, Z_a \rangle$ from $\mathcal{ID}_a$, $\mathcal{ID}_b$ computes $V_b = \frac{1}{x_b} R_a$ and $Q_a = \mathcal{H}(R_a)$. It then verifies that $\langle R_a, U_a, Z_a, V_b, Q_a \rangle$ is indeed an accepting transcript of $\mathcal{ID}_a$'s identification. That is, it checks that $e(Z_a - V_b, P_a) = e(Q_a, Y_a)$ and $e(V_b, V_b) = e(U_a, P_b)$. If both tests pass $\mathcal{ID}_b$ accepts $\mathcal{ID}_a$'s authentication. If $\mathcal{ID}_b$ decides to continue with the process it generates $Q_b \leftarrow \mathbb{G}_1$, computes $Z_b = V_b + x_b Q_b$ and sends $\langle Z_b, Q_b \rangle$ to $\mathcal{ID}_a$ as its response. It also keeps V_b as the shared key.

3. $\mathcal{ID}_a$ accepts $\mathcal{ID}_b$'s authentication if $e(Z_b - r_aP_b, P_b) = e(Q_b, Y_b)$ and keeps r_aP_b as the shared key.

We claim that in the second step, $\mathcal{ID}_b$ will accept if and only if $\mathcal{ID}_a$ knows r_a and x_a. To see this, first note that $\langle Z_a, Q_a\rangle$ is a zero knowledge identification proof of $\mathcal{ID}_a$. Due to this, there is no guarantee that the proof was generated by $\mathcal{ID}_a$ (since it could also have been efficiently simulated according to section 5). However, observe that if this protocol is simulated, the resulting Q_a will almost certainly be random. A simulator cannot choose a predetermined value of Q_a since there appears to be no way to output an accepting configuration for a specific Q_a without knowledge of x_a. The use of the hash function additionally ensures that the simulator did not have control even over the random coin tosses r_a. Hence, for this particular instance, we can safely assume that the simulation must have been carried out by $\mathcal{ID}_a$. The second and third steps of the protocol involve the identification of $\mathcal{ID}_b$ to $\mathcal{ID}_a$ using the protocol of section 5 keeping r_a as the random coin tosses of verifier $\mathcal{ID}_a$. We feel that this brief analysis is sufficient to understand the security of the protocol.

9.5 Deniable Signcryption

In section 9.1, we presented a scheme that allows a user $\mathcal{ID}_a$ to send authenticated ciphertexts to another user (say $\mathcal{ID}_b$). We note that it is also possible to send a encrypted messages to $\mathcal{ID}_b$ such that it can be later denied by $\mathcal{ID}_a$. In this scenario, however, the public key of $\mathcal{ID}_b$ must be involved. As before we assume that the private key of $\mathcal{ID}_a$ is x_a corresponding to the public key x_aP_a. Also, let x_b be the private key of $\mathcal{ID}_b$ corresponding to the public key x_bP_b (possibly with $P_a = P_b$).

1. At some point $\mathcal{ID}_b$ asks $\mathcal{ID}_a$ to identify itself. To do this it generates $r_b \leftarrow \mathbb{Z}_q$, computes $R_b = r_bY_a$, $U_b = r_b^2P_a$ and sends $\langle R_b, U_b\rangle$ to $\mathcal{ID}_a$.

2. $\mathcal{ID}_a$ computes $V_a = \frac{1}{x_a}R_b$ and accepts the challenge if $e(V_a, V_a) = e(U_b, P_a)$. Using $\mathcal{ID}_b$'s challenge R_b, $\mathcal{ID}_a$ can then encrypt a message $M_a \in \mathbb{G}_1$ for $\mathcal{ID}_b$ as follows: $\mathcal{ID}_a$ generates $r_a \leftarrow \mathbb{Z}_q$, sets $Q_a = M + 2r_aP_b$ and computes $Z_a = V_a + x_aQ_a$ as always. It then computes $T_a = Q_a - r_aP_b$ and $R_a = r_aY_b$ and sends $\langle Z_a, T_a, R_a\rangle$ as its signcrypted message to $\mathcal{ID}_b$

3. On receiving $\langle Z_a, T_a, R_a\rangle$, $\mathcal{ID}_b$ first computes $Q_a = T_a + \frac{1}{x_b}R_a$ and verifies that (Z_a, Q_a) is a valid accepting transcript for $\mathcal{ID}_a$. That is, it checks that $e(Z_a - r_bP_a, P_a) = e(Q_a, Y_a)$. If this check passes $\mathcal{ID}_b$ computes $M_a = T_a - \frac{1}{x_b}R_a$. The zero knowledge identification property ensures that $\mathcal{ID}_b$ will only accept messages that were actually sent by $\mathcal{ID}_a$. Thus, an adversary cannot make it accept any random message as valid.

We will show that the above scheme allows $\mathcal{ID}_a$ to later deny sending the message M_a. Firstly note that $M_a = Q_a - \frac{2}{x_b}R_a$ and $T_a = Q_a - \frac{1}{x_b}R_a$. For any given pair (M_a, Q_a), it is easy to see that $\mathcal{ID}_b$ has the ability to generate $\langle R_a, T_a\rangle$. Due to this, there is no evidence left for $\mathcal{ID}_b$ to prove that these values were actually computed by $\mathcal{ID}_a$.

Also note that $Z_a - r_bP_a = x_aM_a + 2x_ar_aP_b$. Extracting x_aM_a (which would serve as $\mathcal{ID}_a$'s signature on M_a) using this relation is equivalent to computing $x_ar_aP_b$ from r_aP_b and x_aP_a without knowing r_a or x_a. This is a hard problem of the order of the DHP as shown in [8] (cf. aggregate extraction).

9.6 Online Credit Card Payments

In this section, we will present a simple online payment system with some interesting security features. The protocol requires only one certified key. The seller of a product need not provide a certified key to the buyer, effectively remaining anonymous. The seller must produce some identification to the credit card processor or the bank to ensure that the payment is successful. If the buyer notices a disputed transaction on his credit cart statement, he can ask the bank to reveal the identity of the party who received the money. If the transaction is not disputed, the seller can remain completely anonymous. Moreover, we provide the additional advantage of 'single-use' transactions, that is after having successfully processed a payment, the seller cannot later reuse the same information to process another identical payment. We will assume the identification scheme of section 5 where the buyer is $\mathcal{ID}$ and the seller is $\mathcal{S}$. The protocol also involves a third party $\mathcal{B}$ which could be a bank or a credit card processor. The certified public key of $\mathcal{ID}$ is $Y = xP$. This key could itself serve as a credit card number. We also use a cryptographic hash function $\mathcal{H}_1 : \{0,1\}^* \mapsto \mathbb{G}_1$.

1. The buyer $\mathcal{ID}$ begins by visiting the website of $\mathcal{S}$ and initiating a purchase transaction. The details of the transaction are encapsulated in a request $\mathcal{REQ}$. The tuple $\langle \mathcal{ID}, \mathcal{REQ}, Y\rangle$ is sent to $\mathcal{S}$

2. $\mathcal{S}$ generates random $r \leftarrow \mathbb{Z}_q$ and computes $R = rY = rxP$ and $U = r^2P$. It also creates a contract $\mathcal{CON}$ containing the payment amount, transaction date, time and other details (though it will possibly not mention the identity of the seller or the commodity for sale to protect privacy). It sends $\langle \mathcal{CON}, R, U\rangle$ to $\mathcal{ID}$. It is understood that transactions are accepted as valid by the bank only for a short specified deadline (say five minutes) from the time mentioned in the contract.

3. On receiving $\langle \mathcal{CON}, R, U\rangle$, $\mathcal{ID}$ checks that the contract is correct. It then computes $V = \frac{1}{x}U$ and checks that $e(V, V) = e(U, P)$. It this check passes, it computes $Q = \mathcal{H}_1(\mathcal{CON})$, $Z_1 = V + xQ$ and $Z_2 = x\mathcal{H}(Z_1)$. It sends $\langle Z_1, Z_2\rangle$ back to $\mathcal{S}$ and saves $\langle \mathcal{CON}, R\rangle$ in its database until it receives its next credit card statement from the bank. It also keeps a record of $\mathcal{S}$'s reply to the transaction in case a dispute arises.

4. $\mathcal{S}$ computes $Q = \mathcal{H}_1(\mathcal{CON})$ and verifies that $e(Z_1 - rP, P) = e(Q, Y)$ and $e(Z_2, P) = e(\mathcal{H}(Z_1), Y)$. If both checks pass, $\mathcal{S}$ forwards the tuple $\langle Z_1, Z_2, r, Y, \mathcal{ID}, \mathcal{S}, \mathcal{CON}\rangle$ as a payment request to the bank $\mathcal{B}$.

5. On receiving a payment request, $\mathcal{B}$ does the same verification as $\mathcal{S}$; that is, it computes $Q = \mathcal{H}_1(\mathcal{CON})$ and verifies $e(Z_1 - rP, P) = e(Q, Y)$ and $e(Z_2, P) = e(\mathcal{H}(Z_1), Y)$. It also ensures that the $\langle r, \mathcal{ID}\rangle$ pair has not been previously used by checking its database. Finally, the bank checks the date and time specified in $\mathcal{CON}$ and ensures that it is within the specified expiry period (five minutes) of the current time. If all checks pass, $\mathcal{B}$ accepts this transaction, deducts the amount specified from $\mathcal{ID}$'s account, credits that amounts to $\mathcal{S}$'s account, saves the tuple $\langle Z_1, Z_2, \mathcal{ID}, \mathcal{S}, \mathcal{CON}, r\rangle$ in its database and returns `success` to $\mathcal{S}$.

6. The bank's reply is forwarded to the buyer along with a receipt of a successful transaction.

7. If the bank receives another transaction with the same $\langle r, \mathcal{ID}\rangle$ pair in the future, it outputs `failure`. For security reasons, it also saves the corresponding $\langle \mathcal{S}, \mathcal{S}', r, \mathcal{ID}\rangle$ in a *blacklist* where $\mathcal{S}'$ is the identity of the other seller corresponding to the same pair. This blacklist can be used for further investigation if necessary.

8. Sometime in the future, the bank sends $\mathcal{CON}$ in a credit card statement to $\mathcal{ID}$. If some transaction is disputed, $\mathcal{ID}$ reports the corresponding $\langle \mathcal{CON}, R\rangle$ back to the bank, along with some evidence (eg. a transaction receipt with a `failure` response). The bank can easily trace the disputed seller $\mathcal{S}$ using its database after validating that $R = rY$.

9.7 Identity Based Cryptography

Using the primitives for identification of section 5, any smart card user $\mathcal{S}$ can setup an Identity Based Encryption (IBE) scheme with a group of smart cards dynamically. The interesting feature of our scheme is that the private keys can be distributed over an insecure channel. The infrastructure is roughly as follows: messages for a user $\mathcal{ID}$ can be encrypted using the public key $\mathcal{ID}$. The private key for decryption is given out by $\mathcal{S}$ over an insecure public channel but masked using the *signcryption* procedure of 9.1. Only the real user is able to extract the secret key. Before this is done, however, user $\mathcal{ID}$ must first produce some personal authenticating information (like a passport photocopy) that can truly establish the identity and is a one-time requirement. This request for a private key can be sent over an unencrypted channel as long as it can be authenticated; that is, it must be ensured that the person requesting the key for $\mathcal{ID}$ is indeed $\mathcal{ID}$. For our purpose, we assume that a signature is used. The private key of the user

$\mathcal{S}$ who acts as the KGC for this setup is $x \in \mathbb{Z}_q$ and the corresponding public key is $Y = xP \in \mathbb{G}_1$. Let $\mathcal{H}_1 : \{0,1\}^* \mapsto \mathbb{G}_1$ be a cryptographic hash function. The public key of $\mathcal{ID}_i$ is implicitly understood to be $Q_i = \mathcal{H}_1(\mathcal{ID}_i)$

1. All users must have a prior certified public key to authenticate its requests to $\mathcal{S}$. Each user $\mathcal{ID}_i$ generates $r_i \leftarrow \mathbb{Z}_q$ and computes $R_i = r_i Y = r_i xP$ and $U_i = r_i^2 P$. $\mathcal{ID}_i$ then signs R_i using its certified private key and sends $\langle R_i, U_i \rangle$ to $\mathcal{S}$ over an insecure channel.

2. The KGC $\mathcal{S}$ verifies the signatures and thus authenticates the request of users. For each valid request R_i of $\mathcal{ID}_i$, the KGC computes $V_i = \frac{1}{x} R_i$ and verifies that $e(V_i, V_i) = e(U_i, P)$. If this check passes, it computes $Q_i = \mathcal{H}_1(\mathcal{ID}_i)$ and $Z_i = V_i + xQ_i$ and makes each $\langle Z_i, \mathcal{ID}_i \rangle$ tuple public via an insecure channel.

3. If $\mathcal{ID}_i$ knows corresponding r_i, he/she can compute the private key xQ_i after authenticating it using the method described in section 9.1. The encryption/decryption can be done exactly as described in [5] after this step. The zero-knowledge property ensures that only the right user can compute the private key from Z_i.

We briefly describe the identity based encryption scheme here (further details can be obtained from [5]). Let $\mathcal{H}_2 : \mathbb{G}_2 \mapsto \{0,1\}^k$ be a cryptographic hash function. A random k bit message M for $\mathcal{ID}_i$ is encrypted as follows: generate $\alpha \leftarrow \mathbb{Z}_q$, compute $Q_i = \mathcal{H}_1(\mathcal{ID}_i)$, $C_1 = M \oplus \mathcal{H}_2(e(\alpha Q_i, Y))$ and $C_2 = \alpha P$. The ciphertext (C_1, C_2) is sent to $\mathcal{ID}_i$ who decrypts $M = C_1 \oplus \mathcal{H}_2(e(C_2, xQ_i))$.

10 Summary

In this paper, we proposed the notion of *zero knowledge blind identification*. Informally, in such a protocol, an honest prover reveals only one (intended) bit of information to an honest verifier and reveals less than that information to a dishonest verifier. In effect, using our scheme, any user can correctly identify to a random server and a passive adversary cannot learn anything about the outcome of the identification. Hence we coin the term blind identification. This is an additional feature provided by our scheme in contrast to previously proposed schemes. In an intuitive sense, ours is a model of a non-interactive zero knowledge proof since the security of the protocol is independent of the honesty of the verifier. The constructions presented in this paper arise from the work on identity based encryption [5], group signatures [7], aggregate signatures [8], chained signatures [10, 11] and additive zero knowledge proofs [9]. Referring to the definitions of sections 3 and 5, essentially, the security of our protocol relies on the hardness of deciding if $Z = \frac{1}{x}R + xQ$ for given P, xP, Q, Z and R. Although, this is not a well studied hard problem like the DHP, we feel reasonably confident that it is computationally intractable. Additionally, as mentioned in appendix B, if we are willing to sacrifice the zero-knowledge property for a dishonest verifier, it is possible to completely exclude all references to U in the protocol (specifically, for the protocol of section 5, figure 1, the verifier will send only R and the prover will respond with $\langle Z, Q \rangle$ irrespective of whether R was randomly generated or not. The verification condition $e(V, V) = e(U, P)$ in the third step is also excluded and all extensions are modified accordingly)

In section 9, we show how these simple identification primitives can be used for constructing complex mechanisms like key agreement, digital signatures, encryption and signcryption. As a simple application of our smart card scheme, we propose a model for online credit card and cheque transactions. The protocol can be used in conjunction with the Secure Electronic Transaction (SET) specification or in a completely different infrastructure. As some other applications, we mention subliminal identification, designated verifier proofs and multiuser authentication. For optimal security, the primitives for signing are best implemented in a tamperproof chip supporting elliptic curve point addition and doubling operations. As observed, all the verification primitives require one or two pairing computations and deal with public keys only. Consequently, they are not restricted to a secure tamperproof device and can be implemented on faster processors. We refer the reader to [4] for details on constructing the hash functions used here.

References

[1] Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 186–194, London, UK, 1987. Springer-Verlag.

[2] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 1989.

[3] Constantin Popescu. An identification scheme based on the elliptic curve discrete logarithm problem. 2(2):624, 2000.

[4] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *ASIACRYPT '01: Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security*, pages 514–532, London, UK, 2001. Springer-Verlag.

[5] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 213–229. Springer-Verlag, 2001.

[6] Paulo S. L. M. Barreto, Hae Yong Kim, Ben Lynn, and Michael Scott. Efficient algorithms for pairing-based cryptosystems. In *CRYPTO '02: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, pages 354–368, London, UK, 2002. Springer-Verlag.

[7] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Advances in Cryptology—CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Berlin: Springer-Verlag, 2004.

[8] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer, 2003.

[9] Amitabh Saxena and Ben Soh. Authenticating mobile agent platforms using signature chaining without trusted third parties. In *Proceedings of The 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE-05)*, pages 282–285, Hong kong, 2005. IEEE computer press.

[10] Amitabh Saxena and Ben Soh. One-way signature chaining: A new paradigm for group cryptosystems and e-commerce. Cryptology ePrint Archive, Report 2005/335, 2005.

[11] Amitabh Saxena and Ben Soh. A mobile agent authentication protocol using signature chaining with bilinear pairings. Cryptology ePrint Archive, Report 2005/272, 2005.

[12] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM J. Comput.*, 25(1):169–192, 1996.

APPENDIX

In this section, we will briefly analyze the security of the single user identification scenario (section 5, figure 1). The security of all other extensions (section 9) follows directly from the security of this single user case. For convenience, we give the protocol once again.

In this protocol, a smart card $\mathcal{ID}$ having secret key $x \in \mathbb{Z}_q$ and public key $Y = xP \in \mathbb{G}_1$ identifies itself to a server $\mathcal{S}$ as follows:

1. $\mathcal{ID}$ starts by claiming to know $x \in \mathbb{Z}_q$, the discrete logarithm of $Y \in \mathbb{G}_1$ to base P.
2. The verifier $\mathcal{S}$ generates $r \leftarrow \mathbb{Z}_q$ and computes challenges $R = rY$ and $U = r^2P$. It makes $\langle R, U \rangle$ public. Typically the challenges should have a very short lifetime.

3. $\mathcal{ID}$ computes $V = \frac{1}{x}R$ and verifies $e(V, V) = e(U, P)$. If this test passes, $\mathcal{ID}$ is ensured that R was indeed randomly generated. $\mathcal{ID}$ generates $Q \leftarrow \mathbb{G}_1$ and computes $Z = V + xQ$. It sends $\langle Z, Q\rangle$ as its proof to $\mathcal{S}$.

4. $\mathcal{S}$ accepts if $e(Z - rP, P) = e(Q, Y)$.

A Soundness

The soundness property requires that the protocol should be robust in the event of any one participant being dishonest. The trivial case of both participants being dishonest is ignored.

A.1 Dishonest Prover

Assume that the verifier is honest; that is, $R = rxP$ and $U = r^2P$. Then the task of a dishonest prover is to output a pair $\langle Z', Q'\rangle$ without knowledge of x such that $e(Z' - rP) = e(Q', xP)$ or in other words $Z' = rP + xQ'$. The linear Diffie-Hellman assumption (section 3, number 5) states that this is infeasible without knowledge of x or r.

A.2 Dishonest Verifier

In this scenario, we assume that $\mathcal{S}$ is a dishonest verifier if either (1) $R = rY$ for some r but $\mathcal{S}$ does not know r or (2) There is no r such that $R = rY$. In the second case, it is easy to see that $R \notin \mathbb{G}_1$. We will consider each case separately.

(a) Case 2 ($R \notin \mathbb{G}_1$): We know that $|\mathbb{G}_1| = |\mathbb{G}_2| = q$ such that q is a (large) prime. Consequently, $qP = 0\ \forall P \in \mathbb{G}_1$ and no element of $\mathbb{G}_1$ (other than 0) has an order $< q$. It is obvious that if R is a valid challenge then $R \in \mathbb{G}_1$. It may seem necessary to verify that this by checking that $qR = 0$. However, observe that the validation condition $e(\frac{1}{x}R, \frac{1}{x}R) = e(U, P)$ of the third step will hold if and only if $R \in \mathbb{G}_1$ *and* $U = r^2P$. Hence, this possibility is ruled out.

(b) Case 1 ($R = rY$): In this case $\mathcal{S}$ does not know r and its goal is to output U corresponding to the tuple $\langle P, Y, R\rangle$ such that $U = r^2P$. This is infeasible due to the extended Diffie-Hellman assumption (see section 3, number 3).

Also note that EDHP $\Rightarrow$ LDHP. This implies that a dishonest verifier has to solve a harder problem than a dishonest prover. Alternatively, if a dishonest prover cannot cheat then it is ensured that a dishonest verifier cannot cheat either. This ensures the dishonest verifier zero-knowledge property.

Remark: The inclusion of U is only necessary for the dishonest verifier zero knowledge property; that is, to ensure that $\mathcal{S}$ actually knows r since otherwise, it may be possible to obtain information about x not obtainable by an honest verifier (for instance, an adversary could obtain $\langle Z, Q\rangle$ such that $Z - \frac{1}{x}R = xQ$ for an R of choice). The use of U ensures that R was indeed randomly generated if the DHP is really hard. Due to this, a dishonest verifier cannot successfully get $\mathcal{ID}$ to accept a challenge as valid. We note that it may be possible to completely exclude U from the above protocol without compromising the security (as long as the prover always ensures that $R \in \mathbb{G}_1$).

B Honest Verifier and Prover Secrecy

We will model the security against a passive adversary $\mathcal{A}$ using the following game. Assume that both $\mathcal{ID}$ and $\mathcal{S}$ are honest and participate in n runs of the identification protocol using the same public key $Y = xP$ (here n is a parameter decided by $\mathcal{A}$). For each protocol run i, the participants behave as follows:

1. $\mathcal{S}$ acts like a probabilistic honest verifier. It generates $r_i \leftarrow \mathbb{Z}_q^*$, $U_0' \leftarrow \mathbb{G}_1$ and $b \leftarrow \{0, 1\}$. It computes $R_i = r_iY$, $U_1' = r_i^2P$ and sets $U_i = U_b'$. Finally, it sends $\langle R_i, U_i\rangle$ to $\mathcal{ID}$.

2. $\mathcal{ID}$ acts like a probabilistic honest prover. It generates $Q, Z'_0 \leftarrow \mathbb{G}_1$ and $c \leftarrow \{0,1\}$. It then computes $Z'_1 = \frac{1}{x}R_i + xQ_i$ and $b = [e(\frac{1}{x}R_i, \frac{1}{x}R_i) \overset{?}{=} e(U_i, P)]$. Finally, it sets $Z_i = Z'_{(b \text{ AND } c)}$ and responds with $\langle Z_i, Q_i \rangle$.

The zero-knowledge property requires that $\mathcal{A}$ must not gain any useful information about the outcome of the protocol for any polynomially bounded n. Firstly, $\mathcal{A}$ should not be able to decide the outcome of the verification condition $e(\frac{1}{x}R, \frac{1}{x}R) = e(U, P)$ of the third step of the protocol. Secondly, $\mathcal{A}$ should be unable to decide the outcome of the verification condition $e(Z - rP) = e(Q, Y)$ of the fourth step of the protocol. In other words, $\mathcal{A}$'s task is to solve one or both of the following problems given $\langle P, xP, r_ixP, U_i, Q_i, Z_i \rangle$ such that $1 \leq i \leq n$:

1. Decide if $U_i \overset{?}{=} r_i^2 P$ for at least one i with probability $> 1/2$.

2. Decide if $Z_i \overset{?}{=} xQ_i + r_iP$ for at least one i with probability $> 1/4$.

First we will consider the case for one protocol run; that is, $n = 1$. We see that the first problem is an instance of the Extended Decisional Diffie-Hellman Problem (EDDHP) (see section 3, number 4) while the second problem is an instance of the Linear Decisional Diffie-Hellman Problem (LDDHP) (see section 3, number 6). The extended decisional Diffie-Hellman and the linear decisional Diffie-Hellman assumptions state that these two instances are independently intractable. However, the two assumptions may not apply when used together (this is analogous to the claim that zero-knowledge is not preserved under parallel composition [12]). This scenario does not affect us because of the following two observations: Firstly, the problem domain (or the information available to the adversary) for both the instances is exactly the same. Due to this no extra information is given out when the two problems are used in conjunction with each other. Secondly, note that while EDHP $\Rightarrow$ LDHP, it is almost certain that LDHP $\not\Rightarrow$ EDHP. It is therefore very likely that LDDHP $\not\Rightarrow$ EDDHP. If we assume this, we can conclude that the protocol is secure in a single run as long as both assumptions hold independently of each other. We can then consider the two separate instances as a single instance of a composition of the two problems (note that this is only a heuristic proof).

Now consider the case when $n > 1$. In this case, it is easy to see that one instance of an intractable decisional problem (i.e. $n = 1$) implies the intractability of *all* instances of the same problem ($n > 1$) due to the (honest verifier) zero knowledge property assuming that the coin tosses of the prover are truly random in each protocol run [12]. In other words, all instances of the problem are equivalent as long as we ensure that no two instances take place simultaneously. This is analogous the the fact that zero-knowledge is preserved in sequential composition (the same analogy cannot be used to prove intractability of composite decisional problem for protocols that are not zero knowledge).