Proving Without Knowing: On Oblivious, Agnostic and Blindfolded Provers

Markus Jakobsson[1] * and Moti Yung[2]

[1] Department of Computer Science and Engineering,
University of California, San Diego, La Jolla, CA 92093.
Email: markus@cs.ucsd.edu.
[2] IBM T. J. Watson Research Center, Yorktown Heights, NY 10598.
Email: moti@watson.ibm.com.

Abstract. We introduce *oblivious* decision proofs and *agnostic* decision proofs. In the former, the prover does not have to know whether the instance is in the language proven or not in order to be able to perform the decision proof; in the latter, the prover cannot even find out this information from interacting in the protocol. The proofs are minimum-knowledge, limiting the knowledge exposed to the verifier as well. We demonstrate an easily distributable oblivious computational minimum-knowledge decision proof protocol for proving validity/invalidity of undeniable signatures. This method, using obliviousness, solves an open problem [6] of practical value: the distributed verification of undeniable signatures. We also present an *agnostic* proof for the same language; an agnostic prover reduces the dissemination of trust in the system; in fact, a prover can be *blindfolded* and not get to learn the input. As part of the agnostic protocol, and perhaps of independent interest, we exhibit an efficient zero-knowledge proof of knowledge (possession) of both a base and an exponent of an element of a finite group (and similar algebraic structures). Finally, we show a perfect agnostic minimum-knowledge decision proof protocol for quadratic residuosity modulo Blum integers.

1 Introduction

The future communication infrastructure seems to require the use of cryptographic provers which will assure that certain communication and computations are valid (e.g., proof of origin and validity of e-money.) If the provers need to know the full information about the instances they prove things about, then a lot of trust has to be put on them. In this paper, we demonstrate that sometimes provers can perform their task even if they only possess partial information, and furthermore, the participation in a proof does not need to give them additional knowledge. Our methods, therefore, will help in reducing trust requirements in sound cryptographic systems. Whereas zero-knowledge proofs restrict the advantage gained by the verifier, here we are interested in a simultaneous restriction of the knowledge gained by the prover *and* the verifier.

* Research supported by NSF YI Award CCR-92-570979, Sloan Research Fellowship BR-3311, and The Royal Swedish Academy of Sciences. Work initiated while visiting Digicash. Part of the work done while visiting IBM T.J. Watson.

RESULTS. A *decision proof* is a proof in which the prover shows a string to be in or not to be in a certain input language. We introduce the notions of *oblivious* and *agnostic* decision proofs. We call a decision proof *oblivious* if the prover does not have to know whether the string in question is in the language or not for the proof to succeed, and *agnostic* if he cannot even find out whether he is proving language membership or non-membership.

Oblivious decision proofs are useful in a distributed setting since the distributed provers do not have to reach consensus (regarding what protocol to use) before the start of the protocol. Apart from being an obvious efficiency improvement, this can reduce the trust one has to put in the individual servers of the prover authority. We illustrate this in the setting of verification/disavowal of undeniable signatures by demonstrating an oblivious distributed (computational) minimum-knowledge protocol for signature verification and disavowal. This is minimum knowledge (i.e., transfers only the single bit of knowledge, indicating whether the string is in the language or not) even if any subset of the confirmers are corrupt. This is an improvement over existing distributed methods [6], where lists of valid signatures are kept to avoid corrupt servers from obtaining signatures on arbitrary messages.

Agnostic decision proofs are extensions of oblivious decision proofs, hiding the result of the calculation from the provers, and can be useful to thwart traffic analysis (e.g., in verifying signatures on e-money, the prover may also be asked to "verify" non-signatures in order to hide an actual flow of valid cash transactions.) We give two examples of such protocols, the first, a computational agnostic decision proof protocol for undeniable signatures, and the second a perfect minimum-knowledge, perfect agnostic protocol for proving quadratic residuosity and non-residuosity modulo Blum integers.

For all of these example protocols, the interaction does not let the prover obtain any more information about the instance than what the *verifier* gets by interacting, i.e., at most (as in the first example) whether the instance is in the language or not. The instance in itself is perfectly blinded from him (except for public knowledge like the instance size). We say that such a protocol is *hiding*. We call a prover in a hiding proof *blindfolded*. Of course, we allow the verifier to send a number of queries to the prover who has no idea about the input. The protocol assures that an honest verifier does not get extra knowledge. A cheating verifier may ask arbitrary questions, but practically speaking, as long as this verifier is invoked on a specific input, he can learn only the decision bit about this input and perhaps a limited number of bits on random choices (which is typically acceptable in the applications we have in mind).

Of possible independent interest is the zero-knowledge proof of knowledge [3, 7, 15] of a base and an exponent of a number, which we use as a subprotocol. Here, we prove the knowledge of a triple (a, b, c) for a given input $(A, B) = (a^c, b^c)$ over certain algebraic domains (like Z_p^* and Z_N^*, the latter when the prover knows the factorization of N.) It uses a novel "value splitting" technique that enables a direct and quite efficient proof.

RELATED WORK. Goldwasser, Micali and Rackoff [11] introduced the notion of a *zero-knowledge proof*, a method that allows one to prove a statement while yielding nothing but the validity of this assertion to the verifier. A *minimum-knowledge transfer of information*, or *minimum-knowledge proof*, introduced by Galil, Haber and Yung [10], extends this notion by allowing the transfer of some information (like language membership or non-membership), but leaking no further information. They call a proof in which both membership and non-membership can be proven a *decision proof*. Decision proofs in which the *verifier* does not learn the predicate were introduced by Feige, Fiat and Shamir [7] in the context of identification. In this paper, though, the objective of the proofs are for the verifier to learn the predicate – without the *prover* necessarily learning it. Thus, we extend and generalize the work of Fujioka, Okamoto and Ohta [9], who presented a computational minimum-knowledge decision proof for undeniable signatures, which in terms of this paper could be called oblivious. However, they make no attempt to efficiently distribute the protocol (i.e., using multi-provers), which may be difficult to do.

Our results are inspired by several previous notions: We combine minimum-knowledge proofs with notions for hiding results from an oracle, studied by Abadi, Feigenbaum and Kilian [1]. In their model, player V wants to know the value $f(x)$ for some x, but lacks the power to compute it. P, on the other hand, has this power, and is willing to help V compute the function, but V wants P to be able to obtain as little information as possible about x and $f(x)$. Their result is of asymmetric nature in the sense that it is not minimum-knowledge w.r.t. the verifier, only the prover. Ours is symmetric in nature and hides information both ways. Another difference is that we do not assume (although we sometimes do allow) the prover to be powerful.

Further work that inspired us is that of Beaver, Feigenbaum, Ostrovsky and Shoup [2, 8], where the above instance hiding is combined with minimum-knowledge proof systems in multi and single prover models. They show conditions on languages which have such protocols. The results are different. First, we allow multi provers and allow them to interact, whereas the provers are required to be totally separated in their model. Second, they require $2(n+1)$ provers to prove a theorem on a string of length n, whereas we allow any number of provers. Furthermore, we concentrate on specific, concrete languages that can be efficiently proven, whereas their results, being much more general, is mostly classification results of existential nature.

Our protocols are related to the basic ideas of Yao [16, 17]. We suggest efficient solutions to two-party partial information games whereas Yao's general methods are inefficient. Yao defined and solved the *millionaires' problem*, giving an efficient method for two participants to compare their riches and decide who is wealthier, without revealing their inputs to each other. More generally, he showed that two participants, one with input x, the other with input y, can calculate any function $f(x, y)$ without revealing their input to each other, assuming factoring is hard. It is important to note that the latter is an existential result and that the calculation may be very inefficient. As part of two of our protocols

we present efficient solutions to what we call the *socialist millionaires' problem* and the *extravagant socialist millionaires' problem*, both closely related to the millionaires' problem.

2 Definitions

Model
We assume the model of interactive Turing machines [11], zero-knowledge [11] and minimum-knowledge [10] proof systems.

Proof of Knowledge (sketch; for a formal definition see [3, 15, 11].)
A proof of knowledge is a proof for which there is a polynomial time witness extractor that succeeds with a non-negligible probability. This witness extractor works by rewinding the prover and providing different verifier transcripts for different partially rewound sessions.

Decision Proof [10]
We say that a protocol $(P, V) = ((P_1, P_2), (V_1, V_2))$ is a *decision proof* for a language L with inputs I and error probability $\delta(k)$, where k is the length of the input, if

1. For any $x \in I$ given as input to (P, V), (P_1, V_1) is a proof system such that V_1 accepts, halting with the correct output $(x \in L)$ with probability at least $1 - \delta(k)$. For any $x \notin I$ given as input to (P, V), (P_2, V_2) is a proof system such that V_2 accepts, halting with the correct output $(x \notin L)$ with probability at least $1 - \delta(k)$.
2. For any incorrect prover (P_1^* resp P_2^*), the probability that the verifier halts with the incorrect output is at most $\delta(k)$.

Next, we define properties limiting the knowledge gained by a prover in an interaction:

Oblivious
A decision proof is *oblivious* if the prover does not need to know the *predicate of the instance* (i.e., its membership bit w.r.t. the input language) in order to perform the protocol; thus, $P_1 = P_2$.

Agnostic
A decision proof is computational (perfect) *agnostic* if a polynomial-time limited (computationally unlimited) prover does not learn the *predicate of the instance* by the interaction. Namely, the probability of such a machine to compute the predicate after interaction is only negligibly better than computing it prior to the interaction.

Hiding
A proof is computational (perfect) hiding if a polynomial-time limited (computationally unlimited) prover obtains no *instance specific information*, except for

possibly the *predicate of the instance*. This implies that a prover only gets access to a modified input (we say that he is *blindfolded*) and the prover obtains no advantage in computing the input (similar to [1].)

Now we define what it means to limit the knowledge gained by a verifier:

Minimum-Knowledge [10]
Let (P, V) be a protocol in which on input x, P calculates and sends to V the value $z = f(x)$ for a given function (predicate) f.The protocol is computational (perfect) minimum-knowledge if there exists a probabilistic polynomial-time machine M_V that, given one-time access to an oracle O for $f(x)$, for each auxiliary input y and any input x, produces transcripts $\{M_V(x, y)\}$ with a distribution indistinguishable from (identical to) that of he real transcripts $\{P(x), V(x, y)\}$.

Note next that the verifier may, for example, send multiple queries (about elements in the language) to the prover. Queries are protocol related challenges about elements in the language answered by the prover. The blinded prover in protocols that are hiding can be fooled to answer queries unrelated to the actual input (by definition, it cannot make the connection of a query to an input). As long as the verifier is honest, it asks queries as needed and we can be sure that our protocol is "honest verifier minimum-knowledge" (see e.g., [13], where assuming one-way permutations exist, such protocols can be transformed into a minimum-knowledge protocols).

All of our protocols that are hiding are "honest verifier minimum-knowledge". The protocols in fact, involve multi-queries (which we may allow the verifier to ask)– we call these protocols **minimum-knowledge w.r.t. multi-queries**. If the k queries are generated honestly, the protocol is minimum-knowledge. Otherwise, we may leak k answers about language elements (a simulator may require k accesses to a language oracle), which in practice may be acceptable (as in our case). In particular, when the verifier is restricted to access the input only– all he can ask are queries related to the input or random ones.

2.1 Uses of the above knowledge restrictions

In a distributed setting, oblivious decision proofs can be very convenient. In Appendix A, we demonstrate a potential weakness of existing, non-oblivious methods for verifying and disavowing undeniable signatures. This problem stems from the fact that there are *two* different protocols, one for verification of a valid undeniable signature and one for disavowal of an invalid undeniable signature. Although both of these are zero-knowledge given the right kind of input (i.e., a valid signature for the former, and an invalid signature for the latter) they leak knowledge if, for example, the protocol for verification is used on an invalid signature. In fact, here the verifier obtains the valid signature on the given message, which is problematic in light of the fact that individual servers in a distributed setting have no way of distinguishing valid signatures from invalid ones, and so, are at risk of using the wrong protocol. Two partial solutions to this problem have been suggested [6]: One is for the confirmers to reach consensus

before engaging in the proof (a method which may allow a cheating confirmer to obtain signatures on arbitrary messages in the same way that a verifier could, if caution is not taken.) The other is for the signer to distribute a list of valid signatures to the confirmers, a solution that has problems contradicting the desired properties of undeniable signatures. The latter method is also not very appealing in a technical sense. Using an oblivious minimum-knowledge decision proof, we can solve these problems in an elegant way: since the confirmers do not have to choose what protocol to use, there is no risk that the wrong protocol is chosen. Our solution is minimum knowledge against a set of corrupt confirmers as well as against a normal verifier. Making protocols agnostic instead of merely oblivious can be useful in preventing traffic analysis. It is possible that a verifier in a scenario related to, e.g., electronic payments, does not want the prover to know whether a signature is valid or not. Making such proofs "*hiding*" at the same time prevents the prover from learning any instance specific information, including what the predicate is. This means that each day, regardless of the actual number of signature validations needed, the prover is going to be probed by a given "upper bound" of validation requests. This will prevent the release of the actual flow of signatures which may be crucial in the context of e-commerce.

3 Oblivious and Agnostic Decision Proofs for Undeniable Signatures

3.1 Undeniable Signatures: a short Exposé

An undeniable signature is a signature that cannot be verified without the cooperation of a prover, who is either the signer or a confirmer assigned by the signer. That is, it is not possible to distinguish between a valid and an invalid undeniable signature unless some trap-door information is known. The only efficient type of undeniable signature to date is the following type, introduced by Chaum and van Antwerpen [4].

An Undeniable Signature:
Let $p = ql + 1$ for primes p and q and an integer l, and let g be a generator of G_p. The signer of an undeniable signature has a private key $x \in Z_q$ and a public key $y = g^x \, mod \, p$. The valid signature on a message hashing to m is $m^x \, mod \, p$.

Assumption 1:
The language of undeniable signatures is not in BPP, i.e., given (m, s, g, y, p) it is hard to decide whether $\alpha = \beta$ for input $y = g^\alpha \, mod \, p$ and $s = m^\beta \, mod \, p$.

Verification and Disavowal of an Undeniable Signature:
An undeniable signature (m, s, g, y, p) is verified by proving that $log_m s = log_g y$, and disavowed by proving that $log_m s \neq log_g y$.

3.2 The Oblivious Protocol

In the following, we show the simplified non-distributed prover version for simplicity of notation and ease of reading, but note that the suggested protocols are trivially distributed using standard secret sharing methods, and that the distribution of the function generation is transparent to the verifier, i.e., he will not have to know that the prover is distributed.

Our solution uses as a subprotocol a protocol that solves a problem we call the *socialist millionaires' problem* and the *extravagant socialist millionaires' problem*, both closely related to the *millionaire's problem* [17]: Two people, P and V, want to compare their riches, x_P and x_V, to see whether x_P equals x_V. V learns (only) whether $x_V = x_P$. In the *socialist millionaires' problem*, P learns nothing about x_V; in the *extravagant socialist millionaires' problem*, V is willing to reveal x_V to P after P has committed to the value x_P to be used in the calculation.

In order to decide an undeniable signature in an oblivious fashion, we can use the following protocol, which is minimum-knowledge (it is minimum-knowledge in its distributed version w.r.t. any set of corrupt servers and verifiers.) In the following, all operations will be modulo p, where applicable.

Oblivious protocol for deciding an undeniable signature:
Input to the prover: x, the secret key of the prover, such that $y = g^x$.
Input to the verifier: A tuple (m, s, g, y, p).
Objective: The verifier wants to learn whether (m, s) is a valid message-signature pair w.r.t. (g, y), i.e., whether $log_m s = log_g y$.
The following subprotocol is repeated k times in parallel:

1. The verifier flips a fair coin $b \in_u \{0, 1\}$ and picks a blinding factor $\rho \in_u Z_q$ uniformly at random. He calculates
$$(\overline{m}, \overline{s}) = \begin{cases} (g^\rho, y^\rho) & \text{if } b = 0 \\ (m^\rho, s^\rho) & \text{if } b = 1, \end{cases}$$
and sends $\overline{m}$ to the prover.
2. The prover calculates the undeniable signature $\hat{s} = \overline{m}^x$ on $\overline{m}$.
3. Using the protocol solving the extravagant socialist millionaires' problem, the verifier and the prover compares $\overline{s}$ and $\hat{s}$; at the end of the calculation, the verifier ACCEPTs or REJECTs, depending on whether he thinks that $\overline{s}$ is equal to $\hat{s}$, or not. Let res be this result. The verifier writes (b, res) on his private tape.

The verifier reads all the outputs (b, res) of the k subprotocols and writes

- "(m, s, g, y, p): VALID SIGNATURE" on his private output tape, and accepts, if the result of any of the subprotocols is $(b, res) = (1, ACCEPT)$,
- "CHEATING PROVER" on his private output tape, and rejects, if the result of any of the subprotocols $(b, res) = (0, REJECT)$ and none is $(1, ACCEPT)$.

– "(m, s, g, y, p): INVALID SIGNATURE" on his private output tape, and accepts, otherwise.

Subprotocol[3] solving the extravagant socialist millionaires' problem:
Input to the prover: $\hat{s} \in G_p$.
Input to the verifier: $\overline{s} \in G_p$.
Objective: The verifier wants to learn whether $\hat{s} = \overline{s}$.

1. The verifier sends $c_{\overline{s}} = commit_C(\overline{s})$ to the prover, where $commit_C$ is an unconditionally secure (and conditionally binding) commitment scheme.
2. The prover picks a blinding factor $\beta \in_u Z_q$ uniformly at random. He calculates $(\hat{S}, G) = (\hat{s}^\beta, g^\beta)$ and sends $(\hat{S}, G)$ to the verifier.
3. The verifier picks $u, v \in_u Z_q$ and calculates $z = \overline{s}^u g^v$ and sends z to the prover.
4. The prover calculates
$$\begin{cases} w = z^\beta \\ c_w = commit_R(w) \end{cases}$$
and sends c_w to the verifier. Here, $commit_R$ is a commitment scheme that is conditionally secure.
5. The verifier sends $(\overline{s}, u, v)$ to the prover and the prover verifies that $z = \overline{s}^u g^v$; if not, then he halts.
6. The prover sends w to the verifier and verifies whether $w = \hat{S}^u G^v$. If this is the case, he outputs ACCEPT and halts, otherwise he outputs REJECT and halts.

Remark 1: The above protocol is minimum-knowledge w.r.t. k-queries, i.e., it can be simulated given k accesses to an oracle. It may appear that a knowledge complexity of k bits releases a lot of information; however, note that one can only use the prover as an oracle for deciding whether undeniable signatures are valid or not, and the protocol leaks no other information. Thus, a cheating verifier can only obtain more bits of knowledge than one by verifying several signatures in the same protocol, which seems harmless. In application where we need to control knowledge tightly, we can transform the protocol to be minimum-knowledge [13].

Remark 2: The knowledge complexity of the above protocol can easily be reduced from k bits to merely one bit, without increasing the probability of success of a cheating prover. This can be done by having the verifier release the blinding factor ρ just before the end of the protocol, so that the prover can make sure that either (g, y) or *one specific* (m, s) was used in each round of the protocol. If this is not so, he will halt. (This can be combined with blindfolding to hide (m, s) from the prover.)

[3] This protocol is very closely related to one known protocol [4] for verification of undeniable signatures. In fact, it combines the normal signing procedure, using a random one-time secret key, with the verification protocol using the same key.

Theorem 1: The oblivious protocol for deciding an undeniable signature is complete, oblivious, sound, and minimum-knowledge w.r.t. k-queries.

3.3 The Agnostic Protocol

The agnostic protocol for decision proofs of undeniable signatures is quite similar to the one for oblivious decision proofs, but instead of making calls to a subprotocol solving the extravagant socialist millionaires' problem, we make calls (with the same input) to a subprotocol solving the *socialist millionaires' problem*, i.e., a protocol for comparing inputs in which the verifier's value is not leaked to a p-time prover. The protocol for solving the socialist millionaires' problem, in turn, is rather similar to that solving the extravagant version:
Instead of committing to and sending $\overline{s}$, the verifier commits to and sends $\overline{s}^{\alpha}$ for a secret $\alpha \in_u Z_q^*$. Moreover, he calculates g^{α} and sends this to the prover, after which he proves that he knows a value $\overline{s}$ such that there is a value α so that $\overline{s}^{\alpha}$ and g^{α} are the values previously sent. This is a proof of knowledge w.r.t. $\overline{s}$. We actually give a more general proof, namely a zero-knowledge proof of knowledge (i.e., possession) of *both* the bases and the common exponent of two inputs. There are assumed hard problems which are expressed this way (perhaps with added constraints); e.g., we use the problem assuming one of the bases is a known generator.

Base-and-Exponent Proof Protocol:
Input to the prover: (a, b, c); Input to the verifier: (a^c, b^c)
Part 1 (proof of knowledge of a, b using *base splitting*):

1. The prover selects $a_0, b_0 \in_u Z_p$ and calculates

$$\begin{cases} a_1 = a{a_0}^{-1} \\ b_1 = b{b_0}^{-1} \\ A_i = {a_i}^c \quad i \in \{0,1\} \\ B_i = {b_i}^c \quad i \in \{0,1\} \\ c_{ai} = commit(a_i) \quad i \in \{0,1\} \\ c_{bi} = commit(b_i) \quad i \in \{0,1\}. \end{cases}$$

   He sends $\{A_i, B_i, c_{ai}, c_{bi}\}_{i=0}^{1}$ to the verifier.
2. The verifier checks that $A = A_0 A_1$ and $B = B_0 B_1$, picks $e_a, e_b \in_u \{0,1\}$ and sends (e_a, e_b) to the prover.
3. The prover sends a_{e_a} and b_{e_b} to the verifier. The verifier checks that the corresponding c_{ae_a} and c_{be_b} are correctly calculated.
4. Using a normal proof [4] of validity of an undeniable signature, the prover proves to the verifier that $log_{a_{e_a}} A_{e_a} = log_{b_{e_b}} B_{e_b}$.

Part two (proof of knowledge of c, using *exponent splitting*):

5. The prover selects $c_0 \in_u Z_q$ and calculates

$$\begin{cases} c_1 = c - c_0 \, mod \, q \\ C_0 = {a_{e_a}}^{c_0} \\ C_1 = {a_{e_a}}^{c_1}. \end{cases}$$

   He sends (C_0, C_1) to the verifier.
6. The verifier checks that $A_{e_a} = C_0 C_1$ and sends a challenge $e \in_u \{0,1\}$ to the prover.
7. The prover sends c_e to the verifier and the verifier checks that $C_e = {a_{e_a}}^{c_e}$.

There are two versions of the above protocol:

- If *commit* is unconditionally secure, then the above is a perfect zero-knowledge argument of knowledge.
- If *commit* is conditionally secure, then the above is a computational zero-knowledge proof of knowledge.

The above protocol is repeated k times (possibly in parallel) to bring down the error probability to $\frac{1}{2^k}$. We show that:

Theorem 2: The base-and-exponent proof protocol is a complete, sound argument of knowledge which is also perfect zero-knowledge (or, alternatively, a proof of knowledge which is also computational zero-knowledge.)

Based on the above theorem, we conclude that:

Theorem 3: The agnostic protocol for deciding an undeniable signature is complete, agnostic, sound, and minimum-knowledge w.r.t. multi-queries.

Remark 3: We note that the exponent splitting idea can be used to prove knowledge of an exponent pair (a, b) for an unconditionally secure commitment scheme $g^a h^b \bmod p$, or equality between two messages (a_0, a_1) committed to for such a scheme.

Let c_0 and c_1 be the commitment values corresponding to two messages a_0 and a_1, and let b_0, b_1 be random strings. We want to prove that $a = a_0 = a_1$, where $c_i = g^{a_i} h^{b_i} \bmod p$ for $i \in \{0,1\}$:

1. Randomly split a into $a_{(0)}$ and $a_{(1)}$, b_i into $b_{i(0)}$ and $b_{i(1)}$ for $i \in \{0,1\}$
   calculate $c_{a_{(0)}} = commit(a_{(0)})$, $c_{a_{(1)}} = commit(a_{(1)})$, and
   send $(g^{a_{(0)}} h^{b_{0(0)}}, g^{a_{(1)}} h^{b_{0(1)}}, g^{a_{(0)}} h^{b_{1(0)}}, g^{a_{(1)}} h^{b_{1(1)}}, c_{a_{(0)}}, c_{a_{(1)}})$ to the receiver.
2. The receiver sends $e \in_u \{0,1\}$ to us.
3. Send $(a_{(e)}, b_{0(e)}, b_{1(e)})$ to the receiver, who verifies that the corresponding values previously sent were correctly calculated.

The above is repeated k times.

4 Acknowledgements

Many thanks to Giovanni Di Crescenzo, Ivan Damgård, Russell Impagliazzo, Yves Moutran and Torben Pedersen for stimulating discussions and important comments on earlier versions, and to Adam Young for reading the current version. Special thanks to Joan Boyar for helpful suggestions and valuable discussions.

References

1. M. Abadi, J. Feigenbaum, and J. Kilian, "On Hiding Information from an Oracle," Journal of Computer and System Sciences, v. 39, n. 1, Aug 1989, pp. 21-50
2. D. Beaver, J. Feigenbaum and V. Shoup, "Hiding Instances in Zero-Knowledge Proof Systems," Crypto '90, pp. 326-338
3. M. Bellare, S. Goldreich, "On Defining Proofs of Knowledge," Crypto '92, pp. 390-420
4. D. Chaum and H. van Antwerpen, "Undeniable Signatures," Crypto '89, pp. 212-216
5. D. Chaum, "Zero-Knowledge Undeniable Signatures," Eurocrypt '90, pp. 458-464
6. D. Chaum, J. Boyar, I. Damgård, M. Jakobsson, T. Pedersen, "Undeniable Signatures: Applications and Theory," manuscript to be submitted.
7. U. Feige, A. Fiat and A. Shamir, "Zero-knowledge Proofs of Identity", Journal of Cryptology, 1988, Vol 1, pp 77-94
8. J. Feigenbaum and R.Ostrovsky, "A Note on One-Prover Instance-Hiding, Zero-Knowledge Proof Systems," Asiacrypt '91.
9. A. Fujioka, T. Okamoto, K. Ohta, "Interactive Bi-Proof Systems and Undeniable Signature Schemes," Eurocrypt '91, pp. 243-256
10. Z. Galil, S. Haber, M. Yung, "Minimum-Knowledge Interactive Proofs for Decision Problems," SIAM Journal of Computing, 1988.
11. S. Goldwasser, S, Micali and C. Rackoff, "The Knowledge Complexity of Interactive Proof Systems", SIAM Journal on Computing, vol. 18, n. 1, Feb. 1989, pp. 186-208
12. T. Okamoto and K. Ohta, "Divertible Zero-Knowledge Interactive Proofs and Commutative Random Self-Reducibility," Eurocrypt '89, pp. 134-149
13. R. Ostrovsky, R. Venkatesan, and M. Yung, "Interactive Hashing Simplifies Zero Knowledge Protocol Design", Eurocrypt '90.
14. T. Pedersen, "Distributed Provers with Applications to Undeniable Signatures," Eurocrypt '91, pp. 221-238
15. M. Tompa and H. Woll, "Random Self-Reducibility and Zero-Knowledge Interactive Proofs of Possession of Information," Proceedings of the 28th FOCS, 1987, pp. 472-482
16. A. Yao, "How to Generate and Exchange Secrets," Proceedings of the 27th FOCS, 1986, pp. 162-167
17. A. Yao, "Protocols for Secure Computations," FOCS '82, pp. 160-164

Appendix A

We look at the verification protocol of [5][4], and show what could happen *if* an attacker who does not know a valid signature for m were still allowed to interact in the protocol for verification of valid signatures. (Even though this can be a distributed protocol w.r.t. the provers, we show the simplified one-prover version for increased clarity.) Alice wants to prove to Bob that $log_m s = log_g y$ for a message pair (m, s) given to her by Bob:

1. The verifier, Bob, chooses $a, b \in_u Z_q$ and calculates $d = m^a g^b$. Bob sends d to Alice.
2. Alice chooses $r \in_u Z_q$ and calculates

$$\begin{cases} w_1 = qg^r \\ w_2 = {w_1}^x \end{cases}$$

   and sends (w_1, w_2) to Bob.
3. Bob sends (a, b) to Alice.
4. Alice verifies that $d = m^a g^b$, and then sends r to Bob, who verifies that

$$\begin{cases} w_1 = dg^r \\ w_2 = s^a y^{b+r}. \end{cases}$$

However, Bob can easily obtain $s = m^x$, i.e., Alice's signature on m, for an arbitrary message m by participating in the protocol above and calculating $s = \left(w_2 y^{-(b+r)}\right)^{1/a} = \left((m^a g^b g^r)^x y^{-(b+r)}\right)^{1/a} = m^x$.

Appendix B

In this section, we exhibit an agnostic, hiding decision protocol for quadratic residuosity and non-residuosity modulo a Blum integer. Here, Alice knows the factorization of N, whereas Bob does not. Bob wants to know whether $\alpha \in Z_N$ belongs to QR_N or QNR_N. There are several zero-knowledge proofs of quadratic residuosity modulo Blum integers (and thus also for quadratic non-residuosity) in the literature, e.g., [11, 7]. We can use any such protocol as a subprotocol in our proof. Consider the following protocol:

1. Bob selects a $\rho \in_u QR_N$ and $c \in_u \{0, 1\}$, and calculates $\tilde{\alpha} = (-1)^c \rho\alpha \bmod N$. He sends $\tilde{\alpha}$ to Alice.
2. Alice proves to Bob , using a standard perfect zero-knowledge proof either that $\tilde{\alpha} \in QR_N$ or $\tilde{\alpha} \in QNR_N$. If $c = 0$, Bob concludes that $\alpha \in QR_N \iff \tilde{\alpha} \in QR_N$; if $c = 1$, Bob concludes that $\alpha \in QR_N \iff \tilde{\alpha} \in QNR_N$.

[4] A similar "attack" can be mounted using the protocol of [14].

It is trivial to see that the above protocol is complete, since Alice knows the factorization of N. Also, the soundness and perfect minimum-knowledge properties follow easily from the soundness and perfect zero-knowledge properties of the standard proof or quadratic residuosity/non-residuosity used. Finally, the protocol is perfect hiding and perfect agnostic since Bob's transcript will be statistically uncorrelated to the quadratic residuosity or quadratic non-residuosity of α (by means of c) and to α itself (by means of ρ.).

The above protocol is similar to the "divertible zero-knowledge" protocol of [12]. In that protocol, a warden manipulates the query to avoid subliminal channels between the prover and verifier, here we allow the verifier to manipulate the queries to hide the input. Protocols in [12] for random self reducible languages can be applied in the current setting.

Appendix C

We sketch the proofs of the previously stated theorems.

Theorem 1: The oblivious protocol for deciding an undeniable signature is complete, oblivious, sound, and minimum-knowledge w.r.t. k-queries.

The protocol is trivially *complete* (as are the other protocols) and *oblivious*. We prove the other properties: First we show that the protocol is minimum-knowledge w.r.t. multi-queries. Let O be an oracle such that for each instance of the subprotocol, it outputs a bit b_i indicating whether the verifier V^* knows that the corresponding message-signature pair $(\overline{m}, \overline{s})$ is valid ($b_i = 1$) or that he does not ($b_i = 0$). We will show how, given access to O, we can generate one set of transcripts T_{valid} and $T_{invalid}$, such that their distributions are indistinguishable to those of real transcripts of the protocol, of valid vs. invalid signatures being verified. Let B be a bit indicating whether to generate a transcript of the type 'valid' ($B = 1$) or "invalid" ($B = 0$).

The following simulation will be performed in parallel for each one of the k parallel sessions:

1. Receive $\overline{m}_i$ and $c_{\overline{s}i}$ from V^*, and b_i from O.
2. Select $\hat{S}_i, G_i \in_u Z_p^*$ and send $(\hat{S}_i, G_i)$ to V^*.
3. Receive z_i from V^*.
4. Select $w_i \in_u Z_p$, calculate $c_{wi} = commit_R(w_i)$, and send c_{wi} to V^*.
5. Receive $(\overline{s}_i, u, v)$. Verify that $c_{\overline{s}i} = commit_C(\overline{s}_i)$ and that $z_i = \overline{s}_i{}^u g^v$; halt if either fails. Rewind V^* to the state right after step 1 above.
6. Let
$$\begin{cases} \hat{s}_i = \overline{s}_i \text{ if } b_i = 1 \vee B = 1 \\ \hat{s}_i \in_u G_p \text{ otherwise.} \end{cases}$$
Calculate $(\hat{S}_i, G_i) = (\hat{s}_i^{\beta_i}, g^{\beta_i})$ and send $(\hat{S}_i, G_i)$ to V^*.
7. Receive z_i from V^*.

8. Let
$$\begin{cases} w_i = {z_i}^{\beta_i} \\ c_{wi} = commit_R(w_i) \end{cases}$$
and send c_{wi} to V^*.
9. Receive $(\overline{s}_i, u, v)$. Verify that $c_{\overline{s}i} = commit_C(\overline{s}_i)$ and that $z_i = {\overline{s}_i}^u g^v$; halt if either fails.
10. Send w_i to V^*.

We see that the above simulator runs in polynomial time in the size of the input. In the case $b_i = 1 \vee B = 1$, the distribution of the simulated transcript is identical to a real transcript; in the other case, the simulated transcript is indistinguishable from a real transcript according to assumption 1, or this would give us a method of deciding undeniable signatures in p-time without knowledge of the secret key. Thus, the protocol is computational minimum-knowledge.
The oblivious protocol for deciding an undeniable signature is sound, since the verifier will only accept if $w = \hat{S}^u G^v$, where the prover sets $(w, \hat{S}, G)$ after seeing $z = \overline{s}^u g^v$, but without any information about $(\overline{s}, u, v)$. Since both $\overline{s}$ and g are generators of G_p, the verifier can only achieve this with a non-negligible probability if $(w, \hat{S}, G) = (z^\beta, \overline{s}^\beta, G^\beta)$, i.e., the prover needs to know $\overline{s}$. □

Theorem 2: The base-and-exponent proof protocol is a complete, sound and a perfect zero-knowledge argument of knowledge (or a computational zero-knowledge proof of knowledge.)

We show that the base-and-exponent proof protocol is sound.
Part 1: If the prover is able to correctly answer both the challenge $e_a = 0$ and $a_a = 1$, then he can calculate $a = a_0 a_1 \bmod p$. Therefore, and by a similar argument for b, part 1 of the protocol is sound.
Part 2: If the verifier can answer both the challenge $e = 0$ and $e = 1$ then he can calculate $c = c_0 + c_1 \bmod p$, and consequently, part 2 is also sound.
Next, we show that the base-and-exponent proof protocol is either a perfect zero-knowledge argument of knowledge or computational zero-knowledge proof of knowledge (depending on the commitment type used). We will prove this by giving a simulator for the prover:

1. Randomly select numbers
$$\begin{cases} \tilde{e}_a, \tilde{e}_b \in_u \{0,1\} \\ c \in_u Z_q \\ a_0, a_1, b_0, b_1 \in_u Z_p \end{cases}$$
and calculate
$$\begin{cases} A_{\tilde{e}_a} = {a_{\tilde{e}_a}}^c \\ A_{1-\tilde{e}_a} = A/A_{\tilde{e}_a} \\ B_{\tilde{e}_b} = {b_{\tilde{e}_b}}^c \\ B_{1-\tilde{e}_b} = B/B_{\tilde{e}_b} \\ c_{ai} = commit(a_i) \quad i \in \{0,1\} \\ c_{bi} = commit(b_i) \quad i \in \{0,1\}. \end{cases}$$

Send $\{A_i, B_i, c_{ai}, c_{bi}\}_{i=0}^{1}$ to the verifier.
2. Receive (e_a, e_b) and halt if $(e_a, e_b) \neq (\tilde{e}_a, \tilde{e}_b)$.
3. Otherwise, continue with steps 3-7 of the standard protocol.

The distribution seen by the verifier after the first move will be identical to (indistinguishable from) the distribution of real transcripts if the commitment scheme is unconditionally secure, (correspondingly, if *commit* is only conditionally secure). Therefore, the probability that we will not halt is at least 1/4 per try. Since the real protocol is followed in the rest of the steps, it generates an identical distribution to the real protocol, and succeeds with probability 1, the protocol will be a perfect zero-knowledge argument when *commit* is unconditionally secure (and conditionally binding), and a computational zero-knowledge proof when *commit* is conditionally secure (and unconditionally binding). □

Theorem 3: The agnostic protocol for deciding an undeniable signature is a complete, agnostic and sound proof, which is minimum-knowledge w.r.t. multi-queries.

First we show that the protocol is indeed agnostic. Assuming that the verifier selects $b = 1$ (the other case is not interesting w.r.t. knowledge leaked from the verifier), the prover will get $(m^\rho, s^{\rho\alpha}, g^\alpha)$ from the verifier, along with a proof that $log_{s^\rho} s^{\rho\alpha} = log_g g^\alpha$. According to Theorem 2, this is a perfect zero-knowledge argument, and only reveals $s^{\rho\alpha}$ and g^α, which are public anyway. Since the prover knows x, he can calculate the triple $(m^\rho, (s^{\rho\alpha})^{1/x}, g^\alpha)$, which, if $s = m^x$, equals $(m^\rho, m^{\rho\alpha}, g^\alpha)$. Assume for the sake of a contradiction that the prover will be able to decide whether $s = m^x$ with a non-negligible probability, after seeing the above mentioned transcript.

According to the Diffie-Hellman assumption, it is not possible to calculate $m^{\rho\alpha}$ given only m^ρ, m^α and m. Therefore, it is also not possible to calculate this given m^ρ, g^α, m and g, or we could set $g = m^\beta$ for a random β and get a contradiction. Thus, the prover will not be able to calculate $m^{\rho\alpha}$. If he were able to decide whether $s = m^x$, then when this equation holds, he could calculate $m^{\rho\alpha}$ simply using $(s^{\rho\alpha})^{1/x}$, which would be a contradiction. We conclude that a p-time prover cannot decide whether he is proving validity or invalidity if the Diffie-Hellman assumption holds.

Soundness is shown analogously to the proof of Theorem 1. Next, we show that the protocol is minimum-knowledge w.r.t. multi-queries. Since the base-and-exponent proof is an argument of knowledge (possession) w.r.t. the base, there is an extractor of the witness, i.e., of $\overline{s}_i$, and this reduces the proof to the proof of Theorem 2, which shows that, given $\overline{s}_i$, the corresponding protocol is minimum-knowledge w.r.t. multi-queries. □