Zero-Knowledge Proofs and Applications

Abstract. A zero-knowledge proof is an interactive proof with the additional property that the verifier learns nothing beyond the correctness of the statement being proved. The theory of zero-knowledge proofs is beautiful and rich, and is a cornerstone of the foundations of cryptography. In the context of cryptographic protocols, zero-knowledge proofs can be used to enforce "good behavior" by having parties prove that they indeed followed the protocol correctly. These proofs must reveal nothing about the parties' private inputs, and as such must be zero knowledge. Zero-knowledge proofs are often considered an expensive (and somewhat naive) way of enforcing honest behavior, and those who view them in this way consider them to be not very useful when constructing efficient protocols. Although this is true for arbitrary zero-knowledge proofs of $\mathcal{NP}$ statements, there are many languages of interest for which there are extraordinarily efficient zero-knowledge proofs. Indeed, in many cases an efficient zero-knowledge proof is the best way to ensure that malicious parties do not cheat.

Readers may refer to [2, Ch4] for the necessary theoretical background, and for definitions and examples. Here we are interested in achieving efficiency.

1 An Example

We start by introducing a protocol due to Schnorr [3], and later abstract away several properties that will be useful in the sequel.

1.1 The Schnorr Protocol

Let p be a prime, q a prime divisor of $p-1$, and g an element of order q in $\mathbb{Z}_p^*$. Suppose that a prover P has chosen a random $w \leftarrow \mathbb{Z}_q$ and has published $h = g^w \mod p$. Protocol 1 below proposed by Schnorr provides a very efficient way for P to convince V that he knows the unique value $w \in \mathbb{Z}_q$ such that $h = g^w \mod p$.

Protocol 1 (Schnorr's Protocol for Discrete Log)

- **Common input:** The prover P and verifier V both have (p, q, g, h).
- **Private input:** P has a value $w \in \mathbb{Z}_q$ such that $h = g^w \mod p$.
- **The protocol:**
  1. The prover P chooses a random $r \leftarrow \mathbb{Z}_q$ and sends $a = g^r \mod p$ to V.
  2. V chooses a random challenge $e \leftarrow \mathbb{Z}_q$ and sends it to P.
  3. P sends $z = r + ew \mod q$ to V , which checks that $g^z = ah^e \mod p$, that p, q are prime and that g, h have order q, and accepts if and only if this is the case.

1.2 Basic Properties

Let R be a binary relation. That is, $R \subset \{0,1\}^* \times \{0,1\}^*$, where the only restriction is that if $(x, w) \in R$, then the length of w is at most $p(|x|)$, for some polynomial $p(\cdot)$. For some $(x, w) \in R$, we may think of x as an instance of some computational problem, and w as the solution to that instance. We call w a *witness* for x. We also define $R(x) = \{w | \exists x \text{ s.t. } (x, w) \in R\}$.

For example, we could define a relation $\mathcal{R}_{DL}$ for the discrete log problem by

$$\mathcal{R}_{DL} = \{((\mathbb{G}, q, g, h), w) | h = g^w\}$$

where it is to be understood that $\mathbb{G}$ is of order q with generator g, that q is prime and that $w \in \mathbb{Z}_q$. In this case, R contains the entire set of discrete log problems of the type we considered above, together with their solutions.

Completeness. The first property we would like to be satisfied is a correctness requirement, namely, whenever Alice knows a valid witness she should always be able to convince Bob.

Definition 1.1 (Completeness). *If P and V follow the protocol on input x and private input w to P where $(x, w) \in R$, then V always accepts.*

The lemma below says that the Schnorr protocol satisfies completeness.

Lemma 1.2. *The protocol 1 satisfies completeness relative to the relation $\mathcal{R}_{DL}$.*

Proof. $g^z = g^{r+ew} = g^r (g^w)^e = ah^e \mod p$. □

Soundness. The second natural property is that it should be hard to convince Bob of the validity of a false statement. It turns out that this can be defined in several ways; the definition below aims at capturing the requirement that in order to convince the verifier it is necessary to actually know a witness.

Let us look again at the protocol 1 in order to get some intuition. Roughly, if some P^*, having sent a, can answer two *different* challenges e, e' correctly, then this means that it could produce z, z' such that $g^z = ah^e \mod p$ and also

$g^{z'} = ah^{e'} \mod p$. We have that $g^{z-z'} = h^{e-e'} \mod p$. Now by the assumption, $e - e' \neq 0 \mod q$ and so it has a multiplicative inverse modulo q. Since g, h have order q, by raising both sides to this power, we get $h = g^{(z-z')(e-e')^{-1}} \mod p$, and so $w = (z - z')(e - e')^{-1} \mod q$.

So loosely speaking, a cheating prover who does not know x can only be able to answer at most one challenge value correctly, since otherwise the argument we just gave would imply that it was in fact able to compute x after all.

The above discussion leads to the definition of *special soundness*, which is tailored to 3-round interactive protocols.

Definition 1.3 (Special soundness). *There exists a polynomial-time algorithm A that given any x and any pair of accepting transcripts $(a, e, z), (a, e', z')$ for x, where $e \neq e'$, outputs w such that $(x, w) \in R$.*

Lemma 1.4. *The protocol 1 satisfies special soundness.*

Honest Verifier Zero Knowledge. Until now we did not consider how much information a proof leaks on the witness. In fact, note that any NP relation has a trivial interactive proof which consists of transmitting the witness in the clear. We would like to define formally what it means for a proof to reveal nothing beyond its validity. This will (gradually) lead to the concept of zero knowledge.

Definition 1.5 (Honest Verifier Zero Knowledge). *There exists a probabilistic polynomial-time simulator M, which on input x and e outputs a transcript of the form (a, e, z) with the same probability distribution as transcripts between the honest P and V on common input x. Formally, for every x and w such that $(x, w) \in R$ and every $e \in \{0, 1\}^t$ it holds that*

$$\{M(x, e)\} \equiv \{\langle P(x, w), V(x, e)\rangle\}$$

where $M(x, e)$ denotes the output of simulator M upon input x and e, and $\langle P(x, w), V(x, e)\rangle$ denotes the output transcript of an execution between P and V, where P has input (x, w), V has input x, and V's random tape equals e.

Intuitively, the above definition guarantees that *honest* protocol executions do not reveal anything about the witness as their distribution can be emulated without knowing the witness. Sometimes the simulated distribution is not identical to the real ones, but only computationally indistinguishable; this is still fine for applications at the price of a negligible term in the concrete bounds.

Lemma 1.6. *The protocol 1 satisfies perfect HVZK.*

Proof. The simulator $M(h, e)$, to simulator the view of an honest verifier V, simply choose a random $z \leftarrow \mathbb{Z}_q$, compute $a = g^z h^{-e} \mod p$ and output (a, e, z).

1.3 Formal Definitions and Properties

Based on the Schnorr example, we will now formally define some abstract properties for a protocol that capture the essential properties of the example. It turns out that this abstraction holds for a wide class of protocols that are extremely useful when constructing efficient secure protocols.

Protocol 2 (Σ Protocol for Relation R)

- **Common input:** The prover P and verifier V both have x.
- **Private input:** P has a value w such that $(x, w) \in R$.
- **The protocol template:**
  1. P sends V a message a.
  2. V sends P a random t-bit string e.
  3. P sends a reply z, and V decides to accept or reject based on the values (x, a, e, z).

Definition 1.7. *A protocol π is a Σ-protocol for relation R if it is a three-round public-coin protocol of the form in Protocol 2 and the following requirements hold:*

- ***Completeness:*** *If P and V follow the protocol on input x and private input w to P where $(x, w) \in R$, then V always accepts.*
- ***Special soundness:*** *There exists a polynomial-time algorithm A that given any x and any pair of accepting transcripts $(a, e, z), (a, e', z')$ for x, where $e \neq e'$, outputs w such that $(x, w) \in R$.*
- ***Special honest verifier zero knowledge:*** *There exists a probabilistic polynomial-time simulator M, which on input x and e outputs a transcript of the form (a, e, z) with the same probability distribution as transcripts between the honest P and V on common input x. Formally, for every x and w such that $(x, w) \in R$ and every $e \in \{0, 1\}^t$ it holds that*

$$\{M(x, e)\} \equiv \{\langle P(x, w), V(x, e)\rangle\}$$

  where $M(x, e)$ denotes the output of simulator M upon input x and e, and $\langle P(x, w), V(x, e)\rangle$ denotes the output transcript of an execution between P and V, where P has input (x, w), V has input x, and V's random tape equals e.

The value t is called the challenge length.

Before proceeding, we give another example of a useful Σ-protocol. Specifically, we consider the case of proving that a tuple $(\mathbb{G}, q, g, h, u, v)$ is of the Diffie-Hellman form, that is, that there exists a w such that $u = g^w$ and $v = h^w$.

By the decisional Diffie-Hellman assumption, the case where $u = g^w$ and $v = h^w$ for some w is computationally indistinguishable from the case where $u = g^w$ and $h = g^{w'}$ for $w \neq w'$. Thus, a proof of this fact is non-trivial.

Formally, Protocol 3 below is a proof system for the relation

$$\mathcal{R}_{DH} = \{((\mathbb{G}, q, g, h, u, v), w) | g, h \in \mathbb{G} \land u = g^w \land v = h^w\}$$

Protocol 3 ($\varSigma$ Protocol for Diffie-Hellman Tuples)

- **Common input:** The prover P and verifier V both have $(\mathbb{G}, q, g, h, u, v)$.
- **Private input:** P has a value w such that $u = g^w$ and $v = h^w$.
- **The protocol:**
  1. P chooses a random $r \leftarrow \mathbb{Z}_q$ and computes $a = g^r$ and $b = h^r$. It then sends (a, b) to V.
  2. V chooses a random challenge $e \leftarrow \mathbb{Z}_q$ and sends it to P.
  3. P sends $z = r + eq \mod q$ to V, who checks that $g^z = au^e$ and $h^z = bv^e$.

Claim. Protocol 3 is a $\varSigma$-protocol for the relation $\mathcal{R}_{DH}$.

Proof. For completeness, observe that $g^z = g^{r+ew} = g^r \cdot (g^w)^e = au^e$. A similar calculation shows that $h^z = bv^e$.

Regarding special soundness, let $((a, b), e, z)$ and $((a, b), e', z')$ be two accepting transcripts. We have that $g^z = au^e$ and $g^{z'} = au^{e'}$, as well as $h^z = bv^e$ and $h^{z'} = bv^{e'}$. We conclude that $g^{(z-z')/(e-e')} = u$ and $h^{(z-z')/(e-e')} = v$. Therefore let $w = (z - z')/(e - e')$, it holds that $((\mathbb{G}, q, g, h, u, v), w) \in \mathcal{R}_{DH}$ as required.

The simulator of special honest verifier zero knowledge is constructed almost identical to that in Protocol 1 and is therefore omitted. □

Properties of $\varSigma$-protocols. We conclude this section with two important, yet easily verified properties of $\varSigma$-protocols.

Lemma 1.8. *The properties of $\varSigma$-protocols are invariant under parallel repetition. That is, the parallel repetition l times of a $\varSigma$-protocol for R with challenge length t yields a new $\varSigma$-protocol for R with challenge length lt.*

Next we show that the challenge can be set to any arbitrary length.

Lemma 1.9. *If there exists a $\varSigma$-protocol π for R with challenge length t, then there exists a $\varSigma$-protocol π' for R with challenge length t', for any t'.*

2 Proof of Membership & Proof of Knowledge

Loosely speaking, we require that the prover be able to convince the verifier of the validity of true statements, while nobody can fool the verifier into believing false statements. Formally:

Definition 2.1 (Proof of Membership). *A protocol (P, V) is a proof of membership for L_R if the following properties are satisfied:*

- ***Completeness:*** *If P and V follow the protocol on input $x \in L_R$ and private input w to P where $(x, w) \in R$, then V always accepts.*

- ***Soundness:*** *For every $x \notin L_R$ and for every prover P^*, the probability that V accepts after interaction with P^* is negligible.*

Proposition 2.2. *Let π be a Σ-protocol for a relation R with challenge length t. Then π is an interactive proof of membership for L_R with soundness error 2^{-t}.*

Proof. We only show soundness here. Let $x \notin L_R$. We show that no P^* can convince V to accept with probability greater than 2^{-t}, even if P^* is computationally unbounded. Assume by contradiction that P^* can convince V with probability greater than 2^{-t}. This implies that there exists a first message a from P^* and at least two queries e, e' from V that result in accepting transcripts. This is the case because if for every a there exists at most one query e that results in an accepting transcript, then P^* would convince V with probability at most 2^{-t}; i.e., P^* would convince V only if V chose the single query e that P^* can answer, and since $e \leftarrow \{0,1\}^t$ this happens with probability only 2^{-t}.

Observe now that the special soundness property requires that A output a witness w such that $(x, w) \in R$ (implying that $x \in L_R$) when given any pair of accepting transcripts $(a, e, z), (a, e', z')$ with $e \neq e'$. Thus, we conclude that whenever P^* can convince V with probability greater than 2^{-t} it holds that $x \in L_R$ in contradiction to the assumption. $\square$

It should be noted that it is not at all straightforward to define proof of knowledge. Specifically, what does it mean for a machine to “know” or “not know” something? The aim of a proof of knowledge is to enable a prover to convince a verifier that it knows a witness for some statement. Thus, it must be the case that only a machine that “knows” a witness should be able to convince the verifier. The definition below formalizes this by essentially saying that a machine knows a witness if it can be used to efficiently compute it. Stated differently, if a witness can be efficiently computed given access to P^*, then this means that P^* knows the witness.

Definition 2.3. *Let $\kappa : \{0,1\}^* \to [0,1]$ be a function. A protocol (P, V) is a* proof of knowledge *for the relation R with knowledge error κ if the following properties are satisfied:*

- ***Completeness:*** *If P and V follow the protocol on input x and private input w to P where $(x, w) \in R$, then V always accepts.*
- ***Knowledge soundness:*** *There exists a constant $c > 0$ and a probabilistic oracle machine K, called the* knowledge extractor*, such that for every interactive prover function P^* and every $x \in L_R$ the machine K satisfies the following condition. Let $\varepsilon(x)$ be the probability that V accepts on input x after interacting with P^*. If $\varepsilon(x) > \kappa(x)$, then upon input x and oracle access to P^*, the machine K outputs a string w such that $(x, w) \in R$ within an expected number of steps bounded by*

$$\frac{|x|^c}{\varepsilon(x) - \kappa(x)}$$

One can think of the error κ as the probability that one can convince the verifier without knowing a valid w, and the ability to convince the verifier with higher probability means that the prover knows w. Furthermore, the higher the probability that P^* convinces V, the more efficient it is to compute w.

Theorem 2.4. *Let π be a Σ-protocol for a relation R with challenge length t. Then π is a proof of knowledge with knowledge error 2^{-t}.*

Proof. See Appendix A.

3 Witness Indistinguishable and Proving Compound Statements

In this section, we show how basic Σ-protocols can be used to prove compound statements. This is a very useful tool in cryptographic protocols and is a good demonstration of the usefulness of Σ-protocols. It is easy to prove the AND of two statements: simply have the prover prove both in parallel with the verify sending a single challenge e for both proofs. It is easy to see that the result is a Σ-protocol for a compound relation comprised of the AND of two statements.

In contrast, proving the OR of two statements is more involved. Specifically, the problem we wish to solve is as follows. A prover and verifier P and V hold a pair (x_0, x_1) for their common input. The prover P wishes to prove to V that it knows a witness w such that either $(x_0, w) \in R$ or $(x_1, w) \in R$, *without revealing which is the case.* We remark that similar ideas can be used to extend the construction below to prove that k out of n statements are true, again without revealing which; see [4]. In addition, the construction below remains unchanged when the statements are for different relations R_0 and R_1 and the goal is for P to prove that it knows a witness w such that either $(x_0, w) \in R_0$ or $(x_1, w) \in R_1$. We present the case of a single relation for the sake of clarity.

3.1 OR Proofs

Protocol 4 below contains the full details of the OR protocol.

Protocol 4 (OR Protocol for Relation R Based on π)

- **Common input:** The prover P and verifier V both have a pair (x_0, x_1).
- **Private input:** P has a value w and a bit b such that $(x_b, w) \in R$.
- **The protocol:**
  1. P computes the first message a_b in π, using (x_b, w) as input. P chooses e_{1-b} at random and runs the simulator M on input (x_{1-b}, e_{1-b}); let $(a_{1-b}, e_{1-b}, z_{1-b})$ be the output of M. P sends (a_0, a_1) to V.
  2. V chooses a random t-bit string s and sends it to P.
  3. P sets $e_b = s \oplus e_{1-b}$ and computes the answer z_b in π to challenge e_b using (x_b, a_b, e_b, w) as input. P sends (e_0, z_0, e_1, z_1) to V.
  4. V checks that $e_0 \oplus e_1 = s$ and that both transcripts (a_0, e_0, z_0) and (a_1, e_1, z_1) are accepting in π, on inputs x_0 and x_1, respectively.

Let $\mathcal{R}_{OR} = \{((x_0, x_1), w) | (x_0, w) \in R \text{ or } (x_1, w) \in R\}$. Then we have:

Theorem 3.1. *Protocol 4 is a Σ-protocol for the relation $\mathcal{R}_{OR}$. Moreover, for any verifier V^*, the probability distribution over transcripts between P and V^*, where w is such that $(x_b, w) \in R$, is independent of b.*

Proof. Correctness follows immediately.

To verify special soundness, let there be two accepting transcripts $(a_0, a_1, s, e_0, e_1, z_0, z_1)$ and $(a_0, a_1, s', e'_0, e'_1, z'_0, z'_1)$ where $s \neq s'$. Since the transcripts are accepting it holds that $e_0 \oplus e_1 = s$ and $e'_0 \oplus e'_1 = s'$. Since $s \neq s'$ this implies that for some $c \in \{0, 1\}$ it must hold that $e_c \neq e'_c$. By the special soundness of π this implies that from (a_c, e_c, z_c) and (a_c, e'_c, z'_c) it is possible to efficiently compute w such that $(x_c, w) \in R$, implying in turn that $((x_0, x_1), w) \in \mathcal{R}_{OR}$.

Special honest verifier zero-knowledge is immediate: given s, choose e_0 and e_1 at random subject to $s = e_0 \oplus e_1$ and run M twice, once with input (x_0, e_0) and once with input (x_1, e_1).

It remains to prove that the probability distribution over transcripts is independent of b. Let V^* be an arbitrary verifier. Then, observe that the distribution over transcripts between P and V^* can be specified as follows. A transcript is of the form $(a_0, a_1, s, e_0, e_1, z_0, z_1)$, where a_0, a_1 are distributed as an honest prover in π would choose them (this is immediate for a_b and holds for a_{1b} by the perfect honest verifier zero-knowledge of π). Then s has whatever distribution $V^*(x_0, x_1, a_0, a_1)$, and e_0, e_1 are random subject to $s = e_0 \oplus e_1$. Finally, z_0 and z_1 both have whatever distribution the honest prover in π outputs upon the respective transcript prefixes (x_0, a_0, e_0) and (x_1, a_1, e_1). As above, this is immediate for z_b and holds for z_{1b} by the perfect honest verifier zero-knowledge of π. We therefore conclude that the distribution is independent of b. □

The last property is also known as witness indistinguishability (WI). There are several different values of w that a prover may know that would enable it to complete the protocol successfully. However, there is no way that a verifier (even a malicious one) can know which of the possible witnesses the prover knows. This is a first sign that it is possible to obtain security properties that hold for arbitrary verifiers, even when starting from a protocol that is only honest verifier zero knowledge.

Definition 3.2. (P, V) *is witness indistinguishable (WI) over* R *if for any* V^**, any* $x \in Land$ *any* $w_1, w_2 \in R(x)$*,the distributions* $(P(x, w_1), V(x))$ *and* $(P(x, w_2), V(x))$ *are computationally indistinguishable.*

Theorem 3.3. *Any* Σ*-protocol is WI.*

4 Zero-Knowledge from Σ-Protocols

In this section, we show how to construct efficient zero-knowledge protocols from any Σ-protocol. We remark that this can be done in a number of ways. We will first present a construction using any perfectly-hiding commitment scheme. This construction achieves zero knowledge, but is not a proof of knowledge. We will then show what needs to be modified so that the result is a zero-knowledge proof of knowledge.

The zero knowledge property is more tricky to define. For zero knowledge we now think of a possibly cheating verifier V^*. The spirit of the definition is that a proof is zero knowledge if whatever V^* learns, he could have learned by himself without any interaction with P. The idea to formalize this is using the notion of a simulator. That is, we make the following definition:

Definition 4.1 (Zero-knowledge[GMR). *] A protocol* (P, V) *is zero-knowledge if for every polynomial time adversary* V^**, there exists a expected polynomial time simulator* S_{V^*}*, given black-box access to* V^**, such that for every* $x \in L$*, the following two random variables are computationally indistinguishable.*

- $\langle P, V\rangle(x)$
- $S_{V^*}(x)$

Definition 4.2 (Computationally Indistinguishable). *Two random variables* X *and* Y *are computationally indistinguishable, if for every probabilistic polynomial-time algorithm* $D : \{0,1\}^* \to \{0,1\}$*, every* $c > 0$*, and all sufficiently large n's*

$$|\Pr[D(X) = 1] - \Pr[D(Y) = 1]| < \frac{1}{n^c}$$

4.1 A Basic Zero-Knowledge Construction

In order to motivate the construction, observe that the reason why a Σ-protocol is not zero-knowledge for arbitrary verifiers is that a simulator cannot predict the challenge e with probability greater than 2^{-t}. Thus, a verifier that outputs a different essentially random challenge for every first message a cannot be simulated. This problem can be solved by simply having the verifier commit to its challenge e before the execution begins. Let com be a perfectly-hiding commitment protocol for committing to binary strings of length t.

Protocol 5 (Zero-Knowledge Proof for L_R Based on π)

- **Common input:** The prover P and verifier V both have x.
- **Private input:** P has a value w such that $(x, w) \in R$.
- **The protocol:**
  1. V chooses a random t-bit string e and sends the commitment $\mathsf{com}(e)$ to P.
  2. P computes the first message a in π, using (x, w) as input, and sends it to V.
  3. V decommits to e to P.
  4. P verifies the decommitment and aborts if it is not valid. Otherwise, it computes the answer z to challenge e according to the instructions in π, and sends z to V.
  5. V accepts if and only if transcript (a, e, z) is accepting in π on input x.

Theorem 4.3. *If COM is a perfectly-hiding commitment protocol and π is a Σ-protocol for relation R, then Protocol 5 is a zero-knowledge proof for L_R with soundness error 2^{-t}.*

Proof. Completeness is immediate. Soundness follows from Proposition 2.2 and the hiding property of the commitment scheme. Specifically, by the perfect hiding property of the commitment protocol, a cheating prover knows nothing of e before it sends a. Thus, there is only a single challenge that it can answer, and the probability that it equals e is at most 2^{-t}.

In order to prove zero knowledge, we construct a black-box simulator S_{V^*}, as follows:

1. S chooses a random r, invokes the probabilistic polynomial-time verifier $V^*(x, r)$ and receives a commitment.
2. S runs the Σ-protocol honest verifier simulator M on a random $\tilde{e}$ in order to obtain a first message a', and hands it to V^*.
3. S receives back V^*'s decommitment. If it is invalid, S outputs $(x, r, a, \perp)$ and halts. Otherwise, let e be the decommitted value. Then S continues as follows:
   (a) S invokes the simulator M upon input e and obtains back (a, z).
   (b) S hands a to V^* and receives back its decommitment. If the decommitment is to e, then S outputs whatever V^* outputs upon receiving (a, e, z) and halts. If the decommitment is to some $e \neq e'$ then S outputs fail. If the decommitment is not valid, S returns to step 1 and repeats with independent randomness).

We first observe that the computational binding of the commitment scheme ensures that the probability that S outputs fail is at most negligible. Next, we claim that the distribution over the output of S given that it does not output fail is identical to the distribution over real transcripts. This follows from the perfect zero-knowledge property of M.

Formally, let $S_{V^*}(x)$ denote the output of the simulator upon input x, and let $\langle P, V^* \rangle(x)$ denote the output of V^* after a real execution. Then, for any PPT distinguisher D, we have

$$\begin{aligned}
&|\Pr[D(S_{V^*}(x)) = 1] - \Pr[D(\langle P, V^* \rangle(x)) = 1]| \\
= &|\Pr[D(S_{V^*}(x)) = 1 \wedge S_{V^*}(x) \neq \mathsf{fail}] - \Pr[D(\langle P, V^* \rangle(x)) = 1] + \Pr[D(S_{V^*}(x)) = 1 \wedge S_{V^*}(x) = \mathsf{fail}]| \\
\leqslant &|\Pr[D(S_{V^*}(x)) = 1 \wedge S_{V^*}(x) \neq \mathsf{fail}] - \Pr[D(\langle P, V^* \rangle(x)) = 1]| + \Pr[S_{V^*}(x) = \mathsf{fail}]
\end{aligned}$$

Now,

$$\begin{aligned}
&|\Pr[D(S_{V^*}(x)) = 1 \wedge S_{V^*}(x) \neq \mathsf{fail}] - \Pr[D(\langle P, V^* \rangle(x)) = 1]| \\
= &|\Pr[D(S_{V^*}(x)) = 1 | S_{V^*}(x) \neq \mathsf{fail}] \Pr[S_{V^*}(x) \neq \mathsf{fail}] - \Pr[D(\langle P, V^* \rangle(x)) = 1]| \\
= &|\Pr[D(S_{V^*}(x)) = 1 | S_{V^*}(x) \neq \mathsf{fail}] (1 - \Pr[S_{V^*}(x) = \mathsf{fail}]) - \Pr[D(\langle P, V^* \rangle(x)) = 1]| \\
\leqslant &|\Pr[D(S_{V^*}(x)) = 1 | S_{V^*}(x) \neq \mathsf{fail}] - \Pr[D(\langle P, V^* \rangle(x)) = 1]| \\
&+ \Pr[[D(S_{V^*}(x)) = 1 | S_{V^*}(x) \neq \mathsf{fail}] \Pr[S_{V^*}(x) = \mathsf{fail}]
\end{aligned}$$

Combining the above with the fact that $] \Pr[S_{V^*}(x) = \mathsf{fail}]$ is negligible, we have that there exists a negligible function μ such that

$$\begin{aligned}
&|\Pr[D(S_{V^*}(x)) = 1] - \Pr[D(\langle P, V^* \rangle(x)) = 1]| \\
\leqslant &|\Pr[D(S_{V^*}(x)) = 1 | S_{V^*}(x) \neq \mathsf{fail}] - \Pr[D(\langle P, V^* \rangle(x)) = 1]| + \mu(|x|)
\end{aligned}$$

We are now claim that

$$\Pr[D(S_{V^*}(x)) = 1] = \Pr[D(\langle P, V^* \rangle(x)) = 1]$$

In order to see this, note that if the execution terminates with the transcript $(x, r, a, \perp)$, then the distributions are identical to the distribution of a as generated by M. Otherwise, in the first pass S receives a decommitment to e and repeats, generating an independent sample of (a, e, z) from M until this same event repeats. Furthermore, the probability that V^* aborts in a real execution is identical to the probability that it aborts in the simulation. Combining the above together, we have that the distributions are identical. We conclude that

$$|\Pr[D(S_{V^*}(x)) = 1] - \Pr[D(\langle P, V^* \rangle(x)) = 1]| \leqslant \mu(|x|)$$

It remains to show that S runs in expected polynomial time. Let p be the probability that V^* sends a valid decommitment (to any value) upon receiving a from S. The key observation is that since M is a perfect zero-knowledge simulator, it follows that the distribution over a when M is invoked upon x and a random $\tilde{e}$ is identical to the distribution over a when M is invoked upon x and the value e that was revealed in the first pass. This implies that V^* sends a valid decommitment in the rewinding phase with probability exactly p. We stress that S halts as soon as V^* sends a valid decommitment in this stage, irrespective of whether it is to the same e or some $e' \neq e$. Thus, the expected running time of S is

$$\text{poly}(|x|) \cdot \left((1-p) + p \cdot \frac{1}{p}\right) = \text{poly}(|x|)$$

completing the proof. □

Efficient perfectly-hiding commitments The complexity of Protocol 5 depends on the cost of running a perfectly-hiding computationally-binding commitment protocol. We describe the highly efficient Pedersen's commitment scheme that is secure under the discrete log assumption; see Protocol 6.

Protocol 6 (The Pedersen Commitment Protocol)

- **Input:** The committer C and receiver R both hold 1^n, and the committer C has a value $x \in \{0,1\}^n$ interpreted as an integer between 0 and 2^n.
- **The commit phase:**
  1. The receiver R chooses $(\mathbb{G}, q, g)$ where $\mathbb{G}$ is a group of order q with generator g and $q > 2^n$. R then chooses a random $a \leftarrow \mathbb{Z}_q$, computes $\alpha = g^a$ and sends $(\mathbb{G}, q, g, \alpha)$ to C.
  2. The committer C verifies that $\mathbb{G}$ is a group of order q, that g is a generator and that $\alpha \in \mathbb{G}$. Then, it chooses a random $r \leftarrow \mathbb{Z}_q$, computes $c = g^r \cdot \alpha^x$ and sends c to R.
- **The decommit phase:** The committer C sends (r, x) to R, which verifies that $c = g^r \cdot \alpha^x$.

In order to see that the above commitment scheme is perfectly hiding, observe that $c = g^r \cdot \alpha^x = g^{r+ax}$. Now, for every $x' \in \mathbb{Z}_q$ there exists a value r' such that $r' + ax' = r + ax \mod q$, specifically, take $r' = r + ax - ax'$. Thus, since r is chosen randomly, the distribution over commitments to x is identical to the distribution over commitments to x'.

In order to prove computational binding, observe that given two decommitments $(x, r), (x', r')$ to $x, x' \in \mathbb{Z}_q$ where $x \neq x'$, it is possible to compute the discrete log of α, exactly as in Schnorr's protocol. Specifically, by the assumption that both decommitments are valid, it holds that $g^r \alpha^x = g^{r'} \alpha^{x'}$ and so $\alpha = g^{(r-r')/(x'-x)}$.

4.2 Zero-Knowledge Proofs of Knowledge

As we have seen, any Σ-protocol is a proof of knowledge. Unfortunately, however, the zero-knowledge protocol of Protocol 5 appears to not be a proof of knowledge. The reason for this is that an extractor cannot send the prover two different queries $e \neq e'$ for the same a, because it is committed to e before the prover P^* sends a. We solve this problem by using a perfectly-hiding trapdoor commitment scheme instead. Such a commitment scheme has the property that there exists a trapdoor that, when known, enables the committer to generate special commitment values that are distributed exactly like regular commitments, but can be opened to any value in the decommitment phase. Then, in the last step of the protocol, after the verifier has already decommitted to e and so is no longer relevant, the prover can send the trapdoor. Although meaningless in a real proof, this solves the problem described above because the knowledge extractor can obtain the trapdoor and then rewind P^* in order to provide it with different values of e for the same a, as needed to extract.

Let com be a perfectly-hiding trapdoor commitment protocol; and denote the trapdoor by trap. We assume that the receiver in the commitment protocol obtains the trapdoor, and that the committer can efficiently verify that the trapdoor is valid if it receives it later. See Protocol 7 for the details of the construction.

Protocol 7 (ZK Proof of Knowledge for R Based on π)

- **Common input:** The prover P and verifier V both have x.
- **Private input:** P has a value w such that $(x, w) \in R$.
- **The protocol:**
  1. V chooses a random t-bit string e and sends the commitment $\mathsf{com}(e)$ to P.
  2. P computes the first message a in π, using (x, w) as input, and sends it to V.
  3. V decommits to e to P.
  4. P verifies the decommitment and aborts if it is not valid. Otherwise, it computes the answer z to challenge e according to the instructions in π, and sends z and the trapdoor trap to V.
  5. V accepts if and only if the trapdoor trap is valid and the transcript (a, e, z) is accepting in π on input x.

Theorem 4.4. *If com is a perfectly-hiding trapdoor commitment protocol and π is a Σ-protocol for relation R, then Protocol 7 is a zero-knowledge proof of knowledge for R with knowledge error 2^{-t}.*

Proof. The fact that the protocol is zero knowledge follows from the proof that Protocol 5 is zero knowledge. In order to demonstrate that the protocol is a proof of knowledge, we need to construct a knowledge extractor. Such extractor behaves exactly as in theorem 2.4. Specifically, whenever the knowledge extractor K obtains the valid trapdoor trap, K can rewind the entire extraction process again, and sends a special commitment that can be opened to any value at a later time. This enables K to send any challenge e that it wishes, relative to the same prover message a. $\square$

Pedersen's commitment is a trapdoor commitment Observe that if the committer in Protocol 6 knows the exponent a chosen by the receiver to compute α, then it can decommit c to any value it wishes. In particular, C can commit to x by sending $c = g^r \cdot \alpha^x$ and can later decommit to any x' by computing $r' = r + axax'$. This then implies that $r' + ax' = r + ax$ and so $g^r \cdot \alpha^x = c = g^{r'} \cdot \alpha^{x'}$. Stated differently, (r', x') is a valid decommitment to c, and so a is the desired trapdoor.

A Proof of Theorem 2.4.

Completeness is clear by definition. We now prove the validity property; i.e., the existence of an extractor K as described above. Let H be the 0/1-matrix with a row for each possible set of random choices ρ by P^*, and one column for each possible challenge value e. An entry $H_{\rho,e}$ equals 1 if V accepts with this random choice and challenge, and 0 otherwise. Using P^* as a black box while setting its internal random coins to be random and choosing a random challenge, we can probe a random entry in H. Furthermore, by rewinding P^* and reusing the same internal random coins as before, we can probe a random entry in the same row. In this light, our goal is to find two 1s in the same row. Using special soundness the resulting two transcripts provide us with sufficient information to efficiently compute a witness w for x.

Let P^* be a prover that convinces V upon input x with probability $\varepsilon(x)$, and assume that $\varepsilon(x) > 2^{-t}$. This implies that $\varepsilon := \varepsilon(x)$ equals the fraction of 1-entries in H. However, it is important to note that this gives no guarantees about the distribution of 1s in a given row. For instance, there may be some rows with many 1s and others with few 1s (a row with a single 1 cannot be used to find w because it does not give two accepting transcripts). Despite the above, we can make the following observation about the distribution of 1s in H. Define a row to be heavy if it contains a fraction of at least $\varepsilon/2$ ones (there are 2^t entries in a row, and so this means that there are $\varepsilon \cdot 2^{t-1}$ 1s in a heavy row). By a simple counting argument, it holds that more than half of the 1s are located in heavy rows. In order to see this, let H' be the sub-matrix of H consisting of all rows that are not heavy, and denote by h' the total number of entries in H', and by h the total number of entries in H. By the assumption, the number of 1s in H is $h\varepsilon$, and the number of 1s in H' is smaller than $h'\varepsilon/2$. This implies that the number of 1s in heavy rows is greater than

$$h\varepsilon - h'\frac{\varepsilon}{2} \geqslant h\varepsilon - h\frac{\varepsilon}{2} = \frac{h\varepsilon}{2}$$

demonstrating that half of the 1s are indeed in heavy rows.

We first show how the extractor works under the assumption that

$$\varepsilon \geqslant 2^{-t+2}$$

so that a heavy row contains at least two 1s (recall that a heavy row has at least $\varepsilon/2$ 1s, and so if ε is at least 2^{-t+2}, there are at least $2^t\varepsilon/2 \geqslant 2$ 1s in each such row). In this case, we will show below that we can find two 1s in the same row in expected time $O(1/\varepsilon)$. This will be more than sufficient since $1/\varepsilon$ is less than required by the definition, namely $1/(\varepsilon - 2^{-t})$.

Our approach will be to first repeatedly probe H at random, until we find a 1 entry, a "first hit". This happens after an expected number of $1/\varepsilon$ tries, because the fraction of 1s in H is exactly ε. By the observation above, with probability greater than 1/2, the first hit lies in a heavy row. Now, if it does (but note that we cannot check if it does), and if we continue probing at random along this row, the probability of finding another 1 in one attempt equals

$$\frac{2^t\varepsilon/2 - 1}{2^t}$$

This implies that the expected time of K to find the second hit satisfies

$$T = \frac{2^t}{2^t\varepsilon/2 - 1} = \frac{2^t}{\varepsilon/2 \cdot (2^t - 2/\varepsilon)} = \frac{2}{\varepsilon} \cdot \frac{2^t}{2^t - 2/\varepsilon} \leqslant \frac{2}{\varepsilon} \cdot \frac{2^t}{2^t - 2^{t-1}} = \frac{4}{\varepsilon}$$

We therefore conclude that if the extractor's first hit was in a heavy row, then it finds two different accepting transcripts, as required for obtaining the witness, within an expected $O(1/\varepsilon)$ tries. However, it may not be the case that the extractor's first hit was in a heavy row, and in such a case, we may spend too much time finding another 1 (if it exists at all). To remedy this, we include an "emergency break", resulting in the following algorithm (which is still not the final extractor, as we will see below):

1. As above, probe random entries in H until the first 1 is found (the first hit). The row in which this is found is called the current row.
2. Next, start the following two processes in *parallel*, and stop when either one stops:
   - **Process Pr$_1$** Probe random entries in the current row, until another 1 entry is found (the second hit).
   - **Process Pr$_2$** Repeatedly flip a coin that comes out heads with probability ε/d, for some constant d (we show how to choose d below) until you get heads. This can be achieved by probing a random entry in H and choosing a random number out of $1, 2, \cdots, d$. Then, output heads if both the entry in H is a 1 and the number chosen is 1.

We now analyze the expected running time of the above algorithm and its probability of success. Regarding the running time, observe that whenever one of the processes stops the entire algorithm halts. Thus, it suffices to analyze the expected running time of Pr$_2$ and this provides an upper bound on the expected running time of the entire algorithm. Now, since Pr$_2$ halts when the first heads appears, we have that its expected running time is d/ε, which for a constant d equals $O(1/\varepsilon)$ as required. We now show that for an appropriate choice of the constant d, the algorithm succeeds in extracting a witness with probability at least 1/8. Observe that the algorithm only succeeds in extracting a witness if Pr$_1$ finishes first. Therefore, we have to choose d so that Pr$_1$ has enough time to finish before Pr$_2$, if indeed the first hit is in a heavy row.

The probability that Pr2 finishes after exactly k attempts is $\varepsilon/d \cdot (1-\varepsilon/d)^{k-1}$. Using the crude estimate $(1 - \varepsilon/d)^{k-1} \leqslant 1$, we have that the probability that Pr$_2$ finishes within k attempts is less than or equal to $k\varepsilon/d$. Thus, by setting $k = 8/\varepsilon$ and $d = 16$ we have that the probability that Pr$_2$ finishes within $k = 8/\varepsilon$ attempts is less than or equal to 1/2.

We are now ready to show that the algorithm succeeds in extracting a witness with probability at least $1/8$. In order to see this, first recall that if the first hit is in a heavy row, then the expected number of steps of Pr_1 is at most $T = \varepsilon/4$ and so by Markov's inequality, the probability that Pr_1 takes $2T \leqslant 8/\varepsilon$ or more steps is upper-bounded by $1/2$. Stated differently, if the first hit is in a heavy row then with probability at least $1/2$ process Pr_1 concludes in less than $8/\varepsilon$ steps. Since the random variables determining the stopping condition of Pr_1 and Pr_2 are independent, we have that if the first hit is in a heavy row then Pr_1 finishes before Pr_2 with probability at least $1/2 \cdot 1/2 = 1/4$. Next, recall that the first hit is in a heavy row with probability at least $1/2$. We have that the probability that the algorithm succeeds equals at least the probability that the first hit is a heavy row times the probability that Pr_1 finishes before Pr_2, conditioned on the first hit being a heavy row. Thus, the algorithm succeeds with probability $1/2 \cdot 1/4 = 1/8$, as required.

We now describe the actual knowledge extractor (however, still only for the case where $\varepsilon \geqslant 2^{-t+2}$). The knowledge extractor K works by repeating the above algorithm until it succeeds. Since the expected number of repetitions is constant (at most eight), we have that K succeeds in finding a witness in an expected number of steps that is bound by $O(1/\varepsilon)$, as desired.

It still remains to consider the case where $2^{-t} < \varepsilon < 2^{-t+2}$; recall that above we assumed that $\varepsilon \geqslant 2^{-t+2}$ but the extractor K must work whenever $\varepsilon \geqslant 2^{-t}$. We treat this case by a separate algorithm, using the fact that when ε is so small, we actually have enough time to probe an entire row. The knowledge extractor K runs this algorithm in parallel with the above one.

Define δ to be such that $\varepsilon = (1+\delta)2^{-t}$. Note that for the values of ε that we consider here it holds that $0 < \delta < 3$. Let R be the number of rows in H. Then we have that the number of 1s in H is at least $(1+\delta) \cdot R$, where the total number of entries in H equals $R \cdot 2^t$. Since there are only R rows, it follows that at most $R-1$ of the $(1+\delta)R$ ones can be alone in a row, and thus at least δR of them must be in rows with at least two 1s. We call such a row semi-heavy. The algorithm here works as follows:

1. Probe random entries until a 1 is found; call this row the current row;
2. Search the entire current row for another 1 entry. If no such found, then go back to Step 1.

It is clear that the algorithm successfully finds a witness. However, the question is whether it does so in time $O(1/\varepsilon - 2^{-t})$. In order to analyze this, note that the fraction of 1s in semi-heavy rows is $\delta/(1+\delta)$ among all 1s (because the fraction of 1s overall is $1+\delta$ and a δ fraction of these are in semi-heavy rows). In addition, the fraction of 1s in semi-heavy rows is at least $\delta/2^t$ among all entries.

Now, the expected number of probes to find a 1 is

$$\frac{1}{\varepsilon} = \frac{2^t}{1+\delta}$$

Furthermore, by the fact that the fraction of 1s in semi-heavy rows among all entries is $\delta/2^t$, the expected number of probes to find a 1 in a semi-heavy row is $2^t/\delta$. We therefore expect to find a 1 in a semi-heavy row after finding $(1+\delta)/\delta$ 1s. This implies that the expected number of times that the algorithm carries out Steps 1 and 2 equals $(1+\delta)/\delta$. In each such repetition, the expected number of probes in Step 1 equals $2^t/(1+\delta)$. Then, once a 1 is found, the number of probes in Step 2 is exactly 2^t. We therefore have that the expected cost is

$$\frac{1+\delta}{\delta} \cdot \left(\frac{2^t}{1+\delta} + 2^t \right) = 2^t \cdot \frac{2+\delta}{\delta} < 5 \cdot \frac{2^t}{\delta}.$$

However,

$$\frac{1}{\varepsilon - 2^{-t}} = \frac{1}{(1+\delta)2^{-t} - 2^{-t}} = \frac{2^t}{\delta},$$

proving that the algorithm runs in time $O(1/(\varepsilon - 2^{-t}))$, as required. This completes the proof that any Σ-protocol with challenge length t is a proof of knowledge with knowledge error 2^{-t}.

References

1. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In STOC, pages 291–304, 1985.
2. O. Goldreich. Foundations of Cryptography: Volume 1 – Basic Tools. Cambridge University Press, 2001.
3. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In CRYPTO, pages 239–252, 1989.
4. R. Cramer, I. Damgard and B. Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In CRYPTO'94, Springer-Verlag (LNCS 839), pages 174–187, 1994.