

# Logical Composition of Proofs in Full Zero-Knowledge

Andrew Clausen\*

June 24, 2008

## Abstract

This paper studies a classic solution to two problems: (i) constructing efficient zero-knowledge proofs for disjunctions of two assertions and (ii) constructing efficient zero-knowledge proofs out of honest-verifier zero-knowledge proofs. The paper proves that the classic solution works – that the transformation yields a zero-knowledge proof. Previously published work only showed a weaker version of zero-knowledge known as witness indistinguishability (although these results seem to be something of a folk theorem in the cryptography community).

## 1 Introduction

Alice is a director of a chemical company. She is appalled by delays to repairs of a leaky pipe that transports a chemical that kills small children. If Alice leaked the email anonymously, no-one would believe it is real. Alice could convince a journalist, Bob that she knows her private key with a zero-knowledge proof, but then she would lose her anonymity, and risk being fired, assassinated, stood up by James Bond, etc. Can Alice convince Bob that she is a director by convincing Bob that she knows one of the directors' private keys – but without revealing which one?

The whistleblower's problem is an example of a logical composition problem. Given zero-knowledge proofs for a set of assertions  $\{\varphi_1, \dots, \varphi_n\}$ , is it possible to construct a zero-knowledge proof of an arbitrary proposition, such as

$$\varphi_1 \wedge (\varphi_2 \vee \neg\varphi_3)?$$

There are three primitive cases to consider:

- **Conjunction:** This is trivial to implement. To prove  $\psi \wedge \varphi$ , run the proofs of  $\psi$  and then  $\varphi$  sequentially.
- **Disjunction:** We know this is possible, since zero-knowledge proofs exist for all languages in  $\mathcal{NP}$ , which is closed under unions. But it is non-trivial to implement. Simply proving the correct proposition would reveal too much information – namely which assertion is true. The rest of this note describes Cramer et al. [1994]'s technique to prove  $\psi \vee \varphi$  when 3-step polynomial-time-prover ZKPs are available for  $\psi$  and  $\varphi$ . They do not prove this is zero-knowledge. This note contains a proof it is zero-knowledge.
- **Negation:** This is impossible unless  $\mathcal{NP} = \text{co}\mathcal{NP}$ , as efficient provers only exist for languages in  $\mathcal{NP}$ .

This means the techniques described in this note will be able to construct zero-knowledge proofs for monotonic propositions – i.e. propositions that only use the  $\vee$  and  $\wedge$  connectives.

To prove a disjunction  $\psi \vee \varphi$ , the prover will simultaneously execute proofs of  $\psi$  and  $\varphi$ . However, one of the ZKPs will be fake, and the other real. The verifier will not know which is fake, and will only be able to deduce that one of the assertions is true.

A surprising by-product of this procedure is that it can boost honest-verifier zero-knowledge proofs (where conversations are only zero-knowledge if the verifier is “honest”) into full zero-knowledge. A dishonest verifier

---

\*Email: [clausen@econ.upenn.edu](mailto:clausen@econ.upenn.edu). Thanks to Jinsong Tan for reading over various drafts.

might be able to carefully select its challenges to reveal more information in an honest-verifier zero-knowledge proof. But since the transformed proof of  $\varphi \vee \varphi$  allows the prover to choose the challenges (subject to some constraints), the verifier has less freedom in choosing “clever” challenges that leak information.

To implement this partial faking, the prover will use a secret sharing scheme, which is the topic of Section 2. Section 3 defines the 3-step protocols that Cramer et al. [1994]’s construction applies to. Finally, Section 4 describes the disjunction proof, and argues that it is indeed zero-knowledge.

## 2 Secret sharing schemes

Our motivation for using secret sharing is for constructing zero-knowledge proofs for disjunctions of assertions, which is described in the following section. But the original (and more intuitive) motivation for secret sharing schemes is controlling a group of agents’ access to a secret. For example, they can enforce a rule that the nuclear weapon launch codes must only be released with the support of at least two-thirds of the cabinet members. Each member would be given a *secret share* such that the launch code can only be computed if at least two thirds of the shares are available.

More formally, a distributor that knows the secret  $s$  would like to construct  $n$  secret shares  $\{s_i\}_{i=1}^n$ , so that any subset containing at least  $k$  of them can be used to compute  $s$ .

In Shamir [1979]’s secret sharing scheme, the distributor constructs the secret shares as follows:

**Protocol 1** (Secret sharing). *Common input:*  $k, n \in \mathbb{N}$  with  $k \leq n$ .  
*Distributor’s private input:*  $s \in \mathbb{F}$ , where  $\mathbb{F}$  is a finite field with at least  $n + 1$  elements.  
*Output:* secret shares  $\{s_i\}_{i=1}^n$ , so that any  $k$  shares can recover the secret  $s$ .

1. The distributor chooses a random polynomial  $f \in \mathbb{F}[x]$  of degree  $k - 1$  so that  $f(0) = s$ . That is, the distributor stores the secret in the constant term  $a_0 = s$ , and randomly chooses  $a_1, \dots, a_{k-1}$ .
2. The distributor gives each receiver  $i \in \{1, \dots, n\}$  the share  $s_i = (x_i, y_i) = (i, f(i))$ .

Then, any group of agents with at least  $k$  members can reconstruct the secret:

**Protocol 2** (Secret reconstruction). *Input:* any  $k$  of the shares  $\{s_i\}_{i=1}^n = \{(x_i, y_i)\}_{i=1}^n$ .  
*Output:*  $s$ .

1. Use Lagrangian interpolation to construct  $f$  from  $S$ . That is, solve the  $k$  simultaneous equations  $\sum_{j=0}^{k-1} a_j x_i^j = y_i$  for the polynomial coefficients  $a_0, \dots, a_{k-1}$ :
2. Output  $s = f(0) = a_0$ .

If the threshold of  $k$  shares is met, then the secret reconstruction succeeds because every degree  $k - 1$  polynomial is uniquely determined by any  $k$  points it passes through. (While this is a fundamental result for  $\mathbb{R}[x]$ , the generalization to finite fields with at least  $k + 1$  elements is also standard.) Thus,  $k$  points on the polynomial  $f$  contain enough information to recover  $f$  completely, and hence the secret  $s = f(0)$ . Since the equations are all linear, they can be solved with standard finite field arithmetic.

If fewer than  $k$  people reveal their secret shares  $s_i$ , then any choice of  $f(0)$  is consistent with some polynomial in  $\mathbb{F}[x]$ , so no information is revealed about  $s$ .

## 3 Three-step proofs

It will be convenient to work with 3-step proofs of the following form.

**Protocol 3** (3-step schema). *Proves:* knowledge of  $w$  such that  $(x, w) \in R$ .  
*Common input:*  $x$ .  
*Prover’s private input:* a witness  $w$ , where  $(x, w) \in R$ .

1. Offer: The prover randomly picks  $m^1$  independently of  $w$ , and sends it to the verifier.
2. Challenge: The prover picks a challenge  $c \in \mathbb{F}$  uniformly at random from a finite field  $\mathbb{F}$  of fixed size, and sends it to the prover.
3. Response: The prover responds with  $m^2$ , based on  $(w, m^1, c)$ .
4. The verifier checks  $m^2$ , based on  $(x, m^1, c)$ .

For example, the graph isomorphism proof has this form. To prove  $G_1 \cong G_2$ , the prover gives the verifier a random graph  $m^1 = G_3 \cong G_1 \cong G_2$ . The verifier issues a challenge  $c \in \mathbb{F}$ . This maps onto a pre-agreed graph  $G(c) \in \{G_1, G_2\}$ . The prover responds with an isomorphism establishing  $G(c) \cong G_3$ .

The weakest possible definition of almost-zero-knowledge would require no information leak when the verifier picks the challenge randomly. (Full zero-knowledge allows a verifier to pick challenges “cleverly”.)

**Definition 4** (Honest verifier 3-step zero-knowledge proof of knowledge). *An 3-step interactive proof  $(P, V)$  of knowledge of the relation  $R$  is a honest verifier zero-knowledge if every verifier  $V^*$  that uniformly and independently chooses the challenge  $c$  has some simulator  $M$  that with trivial output  $\perp$  with probability less than  $\frac{1}{2}$ , and runs from the protocol  $\{(P(x, w), V(x))\}_{(x, w) \in R}$  are computationally indistinguishable from non-trivial runs from the simulator  $\{M(x)\}_{(x, w) \in R}$ .*

The following soundness condition is a stronger-than-usual requirement for how convincing the prover must be. This definition requires that any prover that is capable of answer two different challenges must be able to compute the witness efficiently.

**Definition 5** (Special Soundness Condition). *A three-step proof of knowledge satisfies the special soundness condition if for any two runs  $(m_1, c, m_2)$  and  $(m_1, c', m'_2)$  on  $(x, w) \in R$ , a witness  $w'$  with  $(x, w') \in R$  can be computed in polynomial time.*

## 4 Zero-knowledge proofs of disjunctions

Cramer et al. [1994] prove  $\varphi \vee \psi$  by providing a genuine proof for one of the assertions, and a fake proof for the other. The prover is forced to answer a difficult challenge for any proposition, but can rig an easy challenge for the remaining proposition. The verifier only knows that one of the challenges was difficult, but does not know which one. The verifier concludes that at least one of the assertions is true.

For example, suppose the whistleblower Alice establishes her identity by proving that two graphs  $G_1$  and  $G_2$  are isomorphic. Her boss uses a different pair of graphs  $H_1$  and  $H_2$ . Alice could prove she is either herself or her boss by proving  $(G_1 \cong G_2) \vee (H_1 \cong H_2)$ . At the end of the proof, the verifier will see a valid transcript of each proof – that is  $(G_3, c, G_c \cong G_3)$  and  $(H_3, c', H_{c'} \cong H_3)$ . But the  $c'$  challenge will be rigged so that Alice can answer it easily.

Cramer et al. [1994] implement this idea using a 2-of-2 secret sharing scheme. The verifier picks a challenge  $c$  to be “shared”, and requires the prover split the challenge into two shares,  $c^\varphi$  and  $c^\psi$  so that all of the shares are needed to reveal the “secret”,  $c$ . Each share gives the challenge in the proof of each proposition. The requirement that the shares reveal  $c$  imposes a one-degree-of-freedom constraint on the shares, leaving the prover to pick one and only one of the challenges.

**Protocol 6** (Disjunctions). *Proves:  $\varphi_1 \vee \varphi_2$  using 3-step proofs for  $\varphi_1$  and  $\varphi_2$ .*

*Common input:  $(x_1, x_2)$ , the common inputs to  $\varphi_1$  and  $\varphi_2$ .*

*Prover’s input: a witness of either assertion,  $w \in R_1(x_1) \cup R_2(x_2)$ .*

1. If  $\varphi_1$  is true, then the prover randomly picks  $m_1^1$  and generates a fake proof of  $\varphi_2$  (using the simulator, which exists if  $\varphi_2$  is honest verifier zero knowledge), consisting of  $(m_2^1, c_2, m_2^2)$ . I omit the (almost) symmetric case when  $\varphi_1$  is false and  $\varphi_2$  is true.
2. The prover sends  $(m_1^1, m_2^1)$  to the verifier.

3. The verifier sends the prover a random challenge  $c$ .
4. The prover has to pick  $(c_1, c_2)$  such that  $c_1 + c_2 = c$ . (This is analagous but not the identical to the secret sharing scheme in Section 2.) If  $\varphi_1$  is true, then the prover has already picked  $c_2$ , so the constraints imply  $c_1 = c - c_2$ . Then the prover computes  $m_1^2$  honestly.
5. The prover sends  $(c_1, c_2, m_1^2, m_2^2)$  to the verifier.
6. The verifier checks that  $c_1 + c_2 = c$  and checks each proof  $(m_i^1, c_i, m_i^2)$ .

The following theorem argues that Protocol 6 is zero-knowledge provided it is construct out of honest-verifier zero-knowledge protocols.

**Theorem 7.** *If  $(P_1, V_1)$  and  $(P_2, V_2)$  are 3-step honest-verifier zero-knowledge proofs of knowledge of  $R_1$  and  $R_2$  and satisfy the special soundness condition, then Protocol 6 is a zero-knowledge proof of knowledge of  $R_1 \cup R_2$ .*

*Proof.* The protocol is complete, since the prover is only required to answer one of the challenges – the remaining response is to a fake challenge generated at the start that is rigged to be trivial.

The protocol is sound, because the proof for each of the assertions is sound, and there are only enough degrees of freedom to rig one of the proofs. For details, see Cramer et al.

Next, we show the protocol is zero-knowledge. As usual, we construct a polynomial time program that simulates any conversation between the prover and verifier. Since the verifier could have generated any conversation transcript without the help of the prover, he did not learn anything from the conversation.

Fix any verifier  $V^*$ , construct the simulator  $M$ :

1. Run the simulators  $M_1$  and  $M_2$ . Call their outputs  $(m_i^1, c_i, m_i^2)$ .
2. If the challenge matches, so that the challenge issued by  $V^*$  on input  $(x, m_1^1, m_2^1)$  equals  $c_1 + c_2$ , then return the output of  $V^*$  and terminate.
3. Otherwise rerun the protocol a maximum of

$$\alpha = \log_{1-1/|\mathbb{F}|} \frac{1}{2}$$

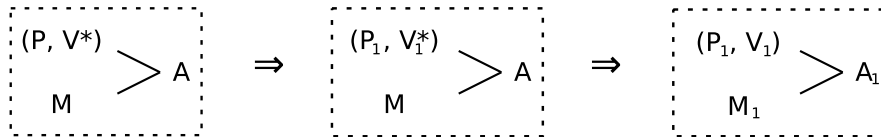
times. If the loop limit is reached, output  $\perp$ .

I claim that  $M$  is a polynomial time simulator of  $(P, V)$ .

First, I will check that  $M$  is polynomial time. Since  $c_1 + c_2$  is uniformly distributed, an iteration succeeds with probability  $1/|\mathbb{F}|$ . If  $|\mathbb{F}|$  is constant, the loop limit  $\alpha$  is constant, so  $M$  is poly time. The choice of  $\alpha$  gives non-trivial termination with probability  $\frac{1}{2}$ .

The rest of the proof establishes that  $M$  is a simulator of  $(P, V)$ . The intuition is: suppose  $P$  is not zero-knowledge, so that a verifier  $V^*$  can learn some information from  $P$ . Then the verifier  $V^*$  can be adapted to  $V_1^*$  (or  $V_2^*$ ), to interact with the prover  $P_1$  instead of  $P$ , and to “dishonestly” extract information from conversations with  $P_1$ . The adapted verifier  $V_1^*$  simulates the proof of  $R_2$ , since  $P_1$  only provides a proof for  $R_1$ . Runs on  $(P_1, V_1^*)$  then produce the same information as  $(P, V^*)$ . Since  $(P_1, V_1^*)$  leaks information, then it must be computationally distinguishable from  $M$ . But this does not immediately contradict the condition that  $(P_1, V_1)$  is honest verifier zero-knowledge, because  $V_1^*$  is not an honest verifier. However,  $V_1^*$  only uses the output of runs from  $(P_1, V_1)$ , which then gives the contradiction.

This diagram summarizes:



Suppose for the sake of contradiction that  $M$  is not a simulator for  $(P, V^*)$ , so that non-trivial runs of  $M$  and  $(P, V^*)$  can be distinguished by some polynomial time Turing machine  $A$ .

If  $A$  can distinguish between  $\{M(x)\}_{(x,w) \in R}$  and  $\{\langle P(x, w), V^*(x) \rangle\}_{(x,w) \in R}$ , then it can also distinguish either

- $\{M(x)\}_{(x,w) \in R_1}$  and  $\{\langle P(x, w), V^*(x) \rangle\}_{(x,w) \in R_1}$ , or
- $\{M(x)\}_{(x,w) \in R_2}$  and  $\{\langle P(x, w), V^*(x) \rangle\}_{(x,w) \in R_2}$ .

Suppose without loss of generality that it can distinguish on the  $R_1$  relation. Let  $V_1^*$  be the hybrid verifier that talks to a real prover  $P_1$  but runs its own simulation for  $R_2$  as follows:

1. Run the honest verifier  $V_1$  with the prover  $P_1$ , and store the output  $(m_1^1, c_1, m_1^2)$ .
2. Run the simulator  $M_2$ , and store its output  $(m_2^1, c_2, m_2^2)$ .
3. If the challenge matches, so that the challenge issued by  $V^*$  on input  $(m_1^1, m_2^1)$  equals  $c_1 + c_2$ , then return the output of  $V^*$  and terminate. Otherwise restart, up to a maximum of  $\alpha$  attempts, outputting  $\perp$  on the final failed attempt.

For convenience, we assume that the prover  $P_1$  loops until the verifier it is paired with terminates.

I claim that non-trivial runs of  $\{\langle P_1, V_1^* \rangle\}_{(x,w) \in R_1}$  are identically distributed to  $\{\langle P, V^* \rangle\}_{(x,w) \in R_1}$ . To see this, observe that the only difference between the two computations is the way  $(c, c_1)$  are chosen. In the hybrid computation  $\langle P_1, V_1^* \rangle$ , the challenge  $c_1$  is chosen randomly, just as the honest verifier  $V_1$  does. But on a successful run,  $c = c_1 + c_2$  matches what  $V^*$  would have chosen after seeing  $(m_1^1, m_2^1)$ . So, by construction, they have the same distribution.

Since  $A$  can distinguish between  $\{M(x)\}_{(x,w) \in R_1}$  and  $\{\langle P(x, w), V^*(x) \rangle\}_{(x,w) \in R_1}$ , it can also distinguish between  $\{M(x)\}_{(x,w) \in R_1}$  and  $\{\langle P_1(x, w), V_1^*(x) \rangle\}_{(x,w) \in R_1}$ . But the only difference between these last two programs is the latter runs an honest verifier ZKP, and the other runs its simulation. But  $V_1^*$  is honest in the sense that it only post-processes the view it has of  $(P_1, V_1)$ . Therefore,  $(P_1, V_1)$  can not be honest verifier zero-knowledge.  $\square$

**Corollary 8.** *If  $(P, V)$  is a 3-step honest verifier zero-knowledge proof of  $R$ , then the proof  $(P', V')$  of  $\varphi \vee \varphi$  as per Protocol 6 is a 3-step zero-knowledge proof of  $R$ .*

## References

- Ronald Cramer, Ivan Damgard, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Proceedings of CRYPTO '94*, pages 174–187. Springer-Verlag, 1994.
- Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.