Zero-Knowledge from Sigma Protocols – An Erratum

Yehuda Lindell[*]

November 13, 2018

Abstract

In this brief note, we correct an error in the proof of Theorem 6.5.2 in [3]. We sincerely thank Joseph Jaeger for pointing out this error, and for suggesting an elegant fix. We have copied the full proof from [3], and will discuss the error and how to fix it. We believe that this error is instructive and therefore have taken the effort to elaborate on it (I am sure that I have personally made this error multiple times in my career). For the sake of clarity, we reproduced the entire relevant subsection from [3] here.

6.5.1 The 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 (say, by applying a pseudorandom function to a) cannot be simulated. This problem can be solved by simply having the verifier commit to its challenge e before the execution begins. This is the methodology used by [2] for the case of computational zero-knowledge. However, the simulation strategy and its analysis is far more simple here because the underlying Σ-protocol has a *perfect* zero-knowledge property (albeit for honest verifiers). Let Com be a perfectly-hiding commitment protocol for committing to binary strings of length t. See Protocol 6.5.1 for the details of the construction.

Theorem 6.5.2 *If Com is a perfectly-hiding commitment protocol and π is a Σ-protocol for relation R, then Protocol 6.5.1 is a zero-knowledge proof for L_R with soundness error 2^{-t}.*

Proof: Completeness is immediate. Soundness follows from Proposition 6.2.3 in [3] 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 $\mathcal{S}$, as follows:

1. $\mathcal{S}$ invokes the polynomial-time verifier V^* upon input x and interacts with it in the commitment protocol.

2. $\mathcal{S}$ 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^*.

[*]Department of Computer Science, Bar-Ilan University, Israel. lindell@biu.ac.il

PROTOCOL 6.5.1 (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 interacts with P via the commitment protocol Com in order to commit to e.
  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.

3. $\mathcal{S}$ receives back V^*'s decommitment. If it is invalid, $\mathcal{S}$ outputs what V^* upon receiving $(x, a', \perp)$ and halts. Otherwise, let e be the decommitted value. Then, $\mathcal{S}$ continues as follows:
   (a) $\mathcal{S}$ invokes the simulator M upon input e and obtains back (a, z).
   (b) $\mathcal{S}$ hands a to V^* and receives back its decommitment. If the decommitment is to e, then $\mathcal{S}$ outputs whatever V^* outputs upon receiving (a, e, z) and halts. If the decommitment is to some $e' \neq e$ then $\mathcal{S}$ outputs fail. If the decommitment is not valid, $\mathcal{S}$ returns to the previous step (and repeats with independent randomness).

We first observe that the computational binding of the commitment scheme ensures that the probability that $\mathcal{S}$ outputs fail is at most negligible. The reduction for this is identical to that in [2] and is omitted (one just needs to use $\mathcal{S}^{V^*}$ to break the computational binding, while truncating the simulation to run in strict polynomial time).

The original proof – conditioning on no fail. In the original proof, we separately considered the case that $\mathcal{S}$ outputs fail and does not output fail. Specifically, we claimed that the distribution over the output of $\mathcal{S}$ given that it does not output fail is close to the distribution over real transcripts by using the perfect zero-knowledge property of M. Formally, let $\mathcal{S}^{V^*}(x)$ 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 distinguisher D, we have

$$\begin{aligned}
&\left|\Pr[D(\mathcal{S}^{V^*}(x)) = 1] - \Pr[D(\langle P, V^* \rangle(x)) = 1]\right| \\
&\quad = \Big|\Pr[D(\mathcal{S}^{V^*}(x)) = 1 \ \wedge\ \mathcal{S}^{V^*}(x) \neq \mathsf{fail}] - \Pr[D(\langle P, V^* \rangle(x)) = 1] \\
&\qquad + \Pr[D(\mathcal{S}^{V^*}(x)) = 1 \ \wedge\ \mathcal{S}^{V^*}(x) = \mathsf{fail}]\Big| \\
&\quad \leq \left|\Pr[D(\mathcal{S}^{V^*}(x)) = 1 \ \wedge\ \mathcal{S}^{V^*}(x) \neq \mathsf{fail}] - \Pr[D(\langle P, V^* \rangle(x)) = 1]\right| \\
&\qquad + \Pr[\mathcal{S}^{V^*}(x) = \mathsf{fail}].
\end{aligned}$$

Now,

$$\begin{aligned}
&\left|\Pr[D(\mathcal{S}^{V^*}(x)) = 1 \ \wedge\ \mathcal{S}^{V^*}(x) \neq \mathsf{fail}] - \Pr[D(\langle P, V^*\rangle(x)) = 1]\right| \\
&= \Big|\Pr[D(\mathcal{S}^{V^*}(x)) = 1 \mid \mathcal{S}^{V^*}(x) \neq \mathsf{fail}] \cdot \Pr[\mathcal{S}^{V^*}(x) \neq \mathsf{fail}] \\
&\qquad - \Pr[D(\langle P, V^*\rangle(x)) = 1]\Big| \\
&= \Big|\Pr[D(\mathcal{S}^{V^*}(x)) = 1 \mid \mathcal{S}^{V^*}(x) \neq \mathsf{fail}] \cdot \Big(1 - \Pr[\mathcal{S}^{V^*}(x) = \mathsf{fail}]\Big) \\
&\qquad - \Pr[D(\langle P, V^*\rangle(x)) = 1]\Big| \\
&\leq \left|\Pr[D(\mathcal{S}^{V^*}(x)) = 1 \mid \mathcal{S}^{V^*}(x) \neq \mathsf{fail}] - \Pr[D(\langle P, V^*\rangle(x)) = 1]\right| \\
&\qquad + \Pr[D(\mathcal{S}^{V^*}(x)) = 1 \mid \mathcal{S}^{V^*}(x) \neq \mathsf{fail}] \cdot \Pr[\mathcal{S}^{V^*}(x) = \mathsf{fail}].
\end{aligned}$$

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

$$\begin{aligned}
&\left|\Pr[D(\mathcal{S}^{V^*}(x)) = 1] - \Pr[D(\langle P, V^*\rangle(x)) = 1]\right| \\
&\leq \left|\Pr[D(\mathcal{S}^{V^*}(x)) = 1 \mid \mathcal{S}^{V^*}(x) \neq \mathsf{fail}] - \Pr[D(\langle P, V^*\rangle(x)) = 1]\right| + \mu(|x|).
\end{aligned}$$

We are now ready to apply the above intuition that the output of $\mathcal{S}$ conditioned on it not outputting fail is close to a real execution transcript.

In the original proof in [3], we claimed that the distribution over the output of $\mathcal{S}$ conditioned on it not outputting fail is *identical* to the output of V^* in a real execution. That is, we claimed that:

$$\Pr[D(\mathcal{S}^{V^*}(x)) = 1 \mid \mathcal{S}^{V^*}(x) \neq \mathsf{fail}] = \Pr[D(\langle P, V^*\rangle(x)) = 1]. \tag{1}$$

This claim is false! There are two reasons for this. First, $\mathcal{S}$ was designed so that an abort due to V^* not opening its commitment correctly happens with the same probability as in the real execution. This can be seen because $\mathcal{S}$ concludes without rewinding if V^* does not provide a valid decommitment, and otherwise it continually rewinds until a valid decommitment is provided again. However, once we *condition* on $\mathcal{S}$ not outputting fail, this skews this probability. Formally, let V^* be a verifier who provides an invalid decommitment with probability p, and in this case outputs the string abort. Furthermore, let V^* be such that it attempts to break the commitment and succeeds in doing so with some negligible probability. In this case, $\mathcal{S}^{V^*}(x)$ outputs fail with some negligible non-zero probability that we denote δ. Thus,

$$\Pr\left[\mathcal{S}^{V^*}(x) = \mathsf{abort}\right] = p \quad \text{and} \quad \Pr\left[\mathcal{S}^{V^*}(x) = \mathsf{fail}\right] = \delta.$$

We have:

$$\begin{aligned}
\Pr\left[\mathcal{S}^{V^*}(x) = \mathsf{abort} \;\middle|\; \mathcal{S}^{V^*}(x) \neq \mathsf{fail}\right] &= \frac{\Pr\left[\mathcal{S}^{V^*}(x) = \mathsf{abort} \ \wedge\ \mathcal{S}^{V^*}(x) \neq \mathsf{fail}\right]}{\Pr\left[\mathcal{S}^{V^*}(x) \neq \mathsf{fail}\right]} \\
&= \frac{\Pr\left[\mathcal{S}^{V^*}(x) = \mathsf{abort}\right]}{\Pr\left[\mathcal{S}^{V^*}(x) \neq \mathsf{fail}\right]} \\
&= \frac{p}{1-\delta} \\
&> p.
\end{aligned}$$

Since the probability that V^* outputs abort in a real execution is p, we have that Eq. (1) does *not* hold for this V^*.

Second, when conditioning on $\mathcal{S}^{V^*}(x)$ not outputting fail, we actually remove events that can happen in the real execution. This also skews the probability, as we show with the following concrete V^*. First, we assume that there is some negligible probability $\epsilon = \epsilon(n)$ for which the binding of the commitment can be completely broken, where the probability is over the randomness used by V^* to try to break it. (For example, if the perfectly hiding commitment is Pedersen, then V^* can try to guess the discrete log by checking if it's random tape is the actual discrete log. If yes, then it can then open a commitment to any value. If the order of the group is q, then V^* breaks the commitment with probability exactly $1/q$.) We construct a malicous verifier as follow. It uses the first bits of its random tape to try to efficiently break the commitment. If it fails, it works honestly. If it succeeds, then it decommits to a random e that is chosen as a function of the first message a received from the prover. (For example, V^* can have a pairwise independent hash function internally, and apply it to a to get e.) In this case, V^* outputs the string broken. Observe that

$$\Pr\left[\langle P, V^*\rangle(x) = \mathsf{broken}\right] = \epsilon.$$

Next, consider the simulator $\mathcal{S}$ working with this V^*. We remark that since V^* never aborts, $\mathcal{S}$ always rewinds V^* exactly once, and either outputs fail or whatever V^* outputs. We differentiate between the first invocation of V^* by $\mathcal{S}$ (before rewinding), and the second invocation (after rewinding). Now, $\mathcal{S}^{V^*}(x)$ outputs broken if and only if V^* broke the commitment, and the e-values chosen in the first and second invocations are the same (since if they are different, $\mathcal{S}^{V^*}(x)$ outputs fail). Since e is chosen as a function of a, this can happen if the a-values chosen in the first and second invocations are the same, or if they are different but e turns out to be the same. Let A be the set of possible a values and assume that a is uniform in A (this is true in typical Sigma protocols). Thus,

$$\Pr\left[\mathcal{S}^{V^*}(x) = \mathsf{broken}\right] = \epsilon \cdot \frac{1}{|A|} + \epsilon \cdot \left(1 - \frac{1}{|A|}\right) \cdot \frac{1}{2^t} = \epsilon \cdot \left(\frac{1}{|A|} + \frac{1}{2^t} - \frac{1}{|A| \cdot 2^t}\right).$$

Next, we compute the probability that $\mathcal{S}^{V^*}(x)$ outputs fail. This happens only if V^* breaks the commitment and sends different e-values. This can only happen if different a-values are sent, and different e-values are chosen. Thus,

$$\Pr\left[\mathcal{S}^{V^*}(x) = \mathsf{fail}\right] = \epsilon \cdot \left(1 - \frac{1}{|A|}\right) \cdot \left(1 - \frac{1}{2^t}\right) = \epsilon \cdot \left(1 - \frac{1}{|A|} - \frac{1}{2^t} + \frac{1}{|A| \cdot 2^t}\right) = \epsilon - \epsilon \cdot \left(\frac{1}{|A|} + \frac{1}{2^t} - \frac{1}{|A| \cdot 2^t}\right).$$

Denoting $z = \left(\frac{1}{|A|} + \frac{1}{2^t} - \frac{1}{|A| \cdot 2^t}\right)$, we have that

$$\begin{aligned}
\Pr\left[\mathcal{S}^{V^*}(x) = \mathsf{broken} \;\middle|\; \mathcal{S}^{V^*}(x) \neq \mathsf{fail}\right] &= \frac{\Pr\left[\mathcal{S}^{V^*}(x) = \mathsf{broken} \;\wedge\; \mathcal{S}^{V^*}(x) \neq \mathsf{fail}\right]}{\Pr[\mathcal{S}^{V^*}(x) = \mathsf{fail}]} \\
&= \frac{\Pr\left[\mathcal{S}^{V^*}(x) = \mathsf{broken}\right]}{\Pr[\mathcal{S}^{V^*}(x) = \mathsf{fail}]} \\
&= \frac{\epsilon \cdot z}{\epsilon - \epsilon \cdot z} \\
&= \epsilon \cdot \frac{z}{1 - z} \\
&\neq \epsilon.
\end{aligned}$$

More discussion on the problems that arise above can be found in [1, Section 2], and we highly recommend reading it.

An alternative proof. The error in our original proof can be fixed by proving that the output of $\mathcal{S}^{V^*}(x)$ is negligibly close to that of a real execution, without using any conditioning. This is preferable since arguing about conditional probabilities is very tricky, as is evident from above. The following elegant alternative proof was suggested by Joseph Jaeger.

We construct a hybrid simulator $\mathcal{H}$ who works in exactly the same way as $\mathcal{S}$, with the exception that in Step 3b, if the decommitment is to some $e' \neq e$ then $\mathcal{H}$ does *not* output fail. Instead, $\mathcal{H}$ returns to step 3a and tries again with independent randomness. (Note that $\mathcal{H}$ does not necessarily run in expected polynomial-time, but this does not matter for the analysis. All we need is that $\mathcal{H}$ halts with probability 1, which holds since V^* decommits to e with non-zero probability.) Since the only difference between $\mathcal{S}$ and $\mathcal{H}$ is when $\mathcal{S}$ would output fail, we have that

$$\left|\Pr\left[D\left(\mathcal{S}^{V^*}(x)\right)=1\right]-\Pr\left[D\left(\mathcal{H}^{V^*}(x)\right)=1\right]\right| \leq \Pr\left[\mathcal{S}^{V^*}(x)=\mathsf{fail}\right]. \tag{2}$$

(This follows from the so-called "difference lemma"; see [4, Lemma 2].) Next, we argue that

$$\Pr\left[D\left(\mathcal{H}^{V^*}(x)\right)=1\right]=\Pr[D(\langle P, V^*\rangle(x))=1]. \tag{3}$$

In order to see that this holds, we distinguish between transcript with invalid and valid decommitments. Then, observe that $\mathcal{H}$ first samples a random transcript. If the transcript contains an invalid decommitment, then it halts and outputs that transcript. Otherwise, it generates a random sample of a transcript with a valid decommitment. This is equivalent to $\mathcal{H}$ first sampling a transcript and checking if it contains an invalid or valid decommitment. Then, in both cases, it re-samples a transcript with the same type of decommitment. This latter procedure clearly provides a transcript that is identical to a real one.

Combining Equations Eq. (2) and Eq. (3), we have that the output distribution generated by the simulator is statistically close to that of the verifier in a real proof.

It remains to show that $\mathcal{S}$ runs in expected polynomial time. Let p be the probability that V^* sends a valid decommitment (to any value) upon receiving a from $\mathcal{S}$. 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 $\mathcal{S}$ 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 $\mathcal{S}$ is

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

completing the proof. ■

References

[1] M. Bellare and P. Rogaway. Code-Based Game-Playing Proofs and the Security of Triple Encryption. In *EUROCRYPT 2006*, Springer (LNCS 4004), pages 409–426, 2006. Full version can be found at https://eprint.iacr.org/2004/331.pdf.

[2] O. Goldreich and A. Kahan. How To Construct Constant-Round Zero-Knowledge Proof Systems for NP. *Journal of Cryptology*, 9(3):167–190, 1996.

[3] C. Hazay and Y. Lindell. *Efficient Secure Two-Party Protocols*, Springer 2010.

[4] V. Shoup. Sequences of Games: A Tool for Taming Complexity in Security Proofs. *Cryptology ePrint Archive: Report 2004/332*, (version from 2006). https://eprint.iacr.org/2004/332.pdf.