1 Zero Knowledge Variants and Results

Recall that a *proof-of-knowledge* (PoK) is a protocol between a prover P and verifier V with the property that there exists a knowledge extractor K which can extract a witness from P in the case that V accepts. In this definition, we assume the prover is "all-powerful", i.e., has no bound on its running time. We can weaken this notion by considering an *argument-of-knowledge* (AoK), which is defined equivalently to a PoK except the prover runs in polynomial time. We will demonstrate a constant-round ZKAoK (due to Feige and Shamir [FS89]) later in this lecture.[1] Using our constant-round ZKAoK protocol, we can achieve a constant-round *coin-tossing* protocol (due to Lindell [Lin03]). Finally, we can combine these results to achieve two-party computation with malicious security from any semi-honest protocol, and thus we get constant-round maliciously-secure two-party computation.

2 Witness Indistinguishability

Before proceeding to our constant-round ZKAoK protocol, we need to discuss the notion of *witness indistinguishability* (WI). A protocol is witness-indistinguishable if a malicious verifier cannot distinguish which witness a prover is using. More formally:

Definition 1 A protocol execution $\langle P, V\rangle$ is *witness-indistinguishable* if for all x, w_1, w_2 such that $(x, w_1), (x, w_2) \in R_L$ and for all polynomial time (malicious) verifiers V^*,

$$\left\{\mathbf{View}^{V^*}_{\langle P(x,w_1),V^*(x)\rangle}(1^k)\right\} \stackrel{c}{\approx} \left\{\mathbf{View}^{V^*}_{\langle P(x,w_2),V^*(x)\rangle}(1^k)\right\}.$$

$\diamondsuit$

Clearly ZK implies WI: A ZK proof implies that there exists a simulator $\mathcal{S}$ such that $\left\{\mathbf{View}^{V^*}_{\langle P(x,w_1),V^*(x)\rangle}(1^k)\right\} \stackrel{c}{\approx} \{\mathcal{S}(x)\}$ and $\left\{\mathbf{View}^{V^*}_{\langle P(x,w_2),V(x)\rangle}(1^k)\right\} \stackrel{c}{\approx} \{\mathcal{S}(x)\}$.

We can also show that WI is preserved under parallel composition (by a standard hybrid argument). This implies the following corollary:

Corollary 1 *k-fold parallel repetition of the ZK proof for graph Hamiltonicity from Lecture 16 is a WIPoK with soundness error 2^{-k}.*

[1] Besides having constant round ZKAoKs, we also have constant round ZKPoKs (due to Goldreich and Kahan [GK96]). However, we do not discuss this result further.

3 Constant-round ZKAoK

We now show the Feige-Shamir protocol for constant-round ZKAoKs [FS89]. Let f be a one-way function.

Feige-Shamir Protocol for Constant-round ZKAoKs

$\underline{P(x, w)}$		$\underline{V(x)}$
	$\xleftarrow{\quad y_1, y_2 \quad}$	$r_1, r_2 \xleftarrow{\$} \{0,1\}^n$ $y_i = f(r_i)$
	$\xleftarrow{\text{WIPoK}(f^{-1}(y_1) \text{ or } f^{-1}(y_2))}$	
	$\xrightarrow{\text{WIPoK}(f^{-1}(y_1) \text{ or } f^{-1}(y_2) \text{ or } x \in L)}$	

Theorem 2 *The above protocol is a ZKAoK.*

Proof We prove this in two steps. We first show that the protocol is a ZK proof, and then we show that it is an AoK.

Claim 3 *The above protocol is a ZK proof.*

Let V^* be a cheating verifier. We construct a simulator $\mathcal{S}$ as follows:

Simulator $\mathcal{S}$ for V^*

1. Receive y_1, y_2 from V^*.
2. Verify the PoK from V^*. If the proof fails, then abort. Otherwise, extract r such that $f(r) \in \{y_1, y_2\}$.
3. Run the final proof as an honest prover would, using witness r.

It is easy to see that this simulator is computationally indistinguishable from the real execution by the WI property.

Claim 4 *The above protocol is an argument-of-knowledge.*

Let P^* be a (polynomial-time) cheating prover, and let K' be the knowledge extractor that exists for the WIPoK of the statement "$f^{-1}(y_i)$ or $f^{-1}(y_2)$ or $x \in L$". We construct a knowledge extractor K as follows:

Knowledge Extractor K

1. Choose $r_1, r_2 \xleftarrow{\$} \{0,1\}^k$ and compute $y_i = f(r_i)$ for $i \in \{1, 2\}$.
2. Run the (first) WIPoK using witness r_2.
3. If P^* succeeds in its WIPoK, then run K' to extract either an r such that $f(r) \in \{y_1, y_2\}$ or a witness w such that $(x, w) \in R_L$.

If we can show that K' does not extract an r such that $f(r) \in \{y_1, y_2\}$ except with negligible probability, then this implies that K successfully extracts a witness w such that $(x, w) \in R_L$ except with negligible probability. Indeed, say K' extracts r with $f(r) \in \{y_1, y_2\}$ with probability p. Let p_1 be the probability that $f(r) = y_1$ and let p_2 be the probability that $f(r) = y_2$. Suppose p_1 is non-negligible. We can turn this into an attack on f as follows: Construct an attacker which, given y_1, chooses $r_2 \xleftarrow{\$} \{0,1\}^k$ and sets $y_2 = f(r_2)$, runs Step 2 of K with r_2 as the witness, and then runs Step 3 of K to extract $f^{-1}(y_1)$. This attack succeeds with probability p_1 and thus p_1 must be negligible.

Now, suppose p_2 is non-negligible. Let $\overline{K}$ be the same as K, except it uses witness r_1 in Step 2 instead of r_2, and let p_2' be the probability that K' extracts r with $f(r) = y_2$ when used by $\overline{K}$. By the WI property, it must be the case that $|p_2' - p_2|$ is negligible. Now, a similar attack to the one described in the previous paragraph shows that p_2' must be negligible. Thus, K' extracts r with $f(r) \in \{y_1, y_2\}$ with negligible probability, completing the proof. ■

4 Constant-round Coin Tossing

We can define the (two-party) coin-tossing functionality, $\mathcal{F}_{\mathbf{ct}}$, as follows:

Functionality $\mathcal{F}_{\mathbf{ct}} \to \{0,1\}^k$

Output: The functionality computes $r \xleftarrow{\$} \{0,1\}^k$ and outputs r to both parties.

Now, consider the following protocol for realizing $\mathcal{F}_{\mathbf{ct}}$:

Coin-tossing Candidate Protocol #1

$$
\begin{array}{rcl}
\underline{P_1} & & \underline{P_2} \\
r_1 \xleftarrow{\$} \{0,1\}^k & \xrightarrow{\quad \mathsf{com}(r_1) \quad} & \\
 & \xleftarrow{\quad r_2 \quad} & r_2 \xleftarrow{\$} \{0,1\}^k \\
\text{Output } r_1 \oplus r_2 & \xrightarrow{r_1, \mathsf{decom}(\mathsf{com}(r_1))} & \text{Output } r_1 \oplus r_2
\end{array}
$$

The problem is that this protocol, while it intuitively *looks* secure, is not simulatable when k is the security parameter (the protocol is in fact secure if k is some small fixed constant, such as 1). Consider the case of a malicious P_2^* who sets r_2 to be some function of the commitment sent by P_1. The simulator is thus unable to fix r_1 such that $r_1 \oplus r_2 = r$ for some uniformly chosen r, since r_2 depends on r_1.

Thus, we modify this protocol by adding ZKAoKs such that P_1 proves knowledge of the committed value, allowing this value to be extracted by the simulator:

Coin-tossing Protocol

$\underline{P_1}$		$\underline{P_2}$
$r_1 \xleftarrow{\$} \{0,1\}^k$	$\xrightarrow{\quad \mathsf{com}(r_1) \quad}$	
	$\xrightarrow{\text{ZKAoK that } r_1 \text{ is the committed value}}$	
	$\xleftarrow{\quad r_2 \quad}$	$r_2 \xleftarrow{\$} \{0,1\}^k$
	$\xrightarrow{\quad r_1 \quad}$	
	$\xrightarrow{\text{ZK argument that initial commitment was to } r_1}$	
Output $r_1 \oplus r_2$		Output $r_1 \oplus r_2$

We now prove that this protocol, due to Lindell [Lin03], securely realizes the $\mathcal{F}_{\mathbf{ct}}$ functionality.

Proof Let P_2^* be a malicious party playing the part of P_2 in the coin-tossing protocol. We construct a simulator $\mathcal{S}$ for P_2^* as follows.

Simulator $\mathcal{S}$ for P_2^*

1. Query $\mathcal{F}_{\mathbf{ct}}$, receiving back r.
2. Compute $\mathsf{com}(0)$ and send it to P_2^*.
3. Simulate the ZKAoK about the validity of the previously sent commitment.
4. Receive r_2 from P_2^*.
5. Send $r_1 = r_2 \oplus r$ to P_2^*.
6. Simulate the ZK argument that the initial commitment was to r_1.

The proof is straightforward but involved; see [Lin03] for the details.

We now show a simulator $\mathcal{S}$ for a malicious P_1^*.

Simulator $\mathcal{S}$ for P_1^*

1. Query $\mathcal{F}_{\mathbf{ct}}$, receiving back r.
2. Receive a commitment from P_1^*.
3. Verify the ZKAoK from P_1^* and extract r_1.
4. Send $r_2 = r_1 \oplus r$ to P_1^*.
5. Receive r_1' from P_1^*.
6. Verify P_1^*'s ZK argument, and thus $r_1' = r_1$ with high probability.

Again, the proof is straightforward but involved; see [Lin03]. ■

References

[FS89] Uriel Feige and Adi Shamir. Zero knowledge proofs of knowledge in two rounds. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 526–544, Santa Barbara, CA, USA, August 20–24, 1989. Springer, Berlin, Germany.

[GK96] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–190, 1996.

[Lin03] Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. *Journal of Cryptology*, 16(3):143–184, June 2003.