ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

HS 2025

Prof. R. Wattenhofer
Ard Kastrati

Computational Thinking
Solutions to Exercise 6 (Cryptography)

1 Zero Knowledge Proofs in Geometry

a) To compute the midpoint of two points A and B, draw the line segment $\overline{AB}$ between A and B, and draw two circles with the radius $\overline{AB}$, one centered around A and the other around B. These circles intersect at two points, C and D. Finally, draw the line segment $\overline{AB}$ between C and D. The intersection of the line segments $\overline{AB}$ and $\overline{CD}$ is the midpoint of A and B.

Bisecting angles is only slightly more difficult. To bisect an angle, first draw a small circle around the tip of the angle which intersects both sides of the angle. Label the points where the circle and the sides of the angle intersect as A and B. Finally, find the midpoint M of A and B, and draw a line segment between M and the tip of the angle. This line segment bisects the angle.[1]

b) The following example is one of the possible protocols. Note that Peggy can send angles to Victor by drawing them on a blank canvas which both of them can see.

ZKP in Geometry

Peggy		**Vic**
knows $\alpha, \beta = 3\alpha$		knows β
create a random angle γ		
construct the angle $\tau = 3\gamma$	send over τ $\longrightarrow$	
	$\longleftarrow$ send over c	choose randomly $c \in \{0, 1\}$
construct $\rho = \gamma + c\alpha$	send over ρ $\longrightarrow$	check $3\rho \stackrel{?}{=} \tau + c\beta$

- **Completeness.** One can easily see that if Peggy is honest and knows α, Vic always accepts. More concretely, in the last step $3\rho = 3(\gamma + c\alpha) = 3\gamma + 3c\alpha = \tau + c\beta$.
- **Soundness.** We show that if Peggy can answer both challenges then she knows α. Assume Peggy can answer both challenges $c \in \{0, 1\}$ correctly, the challenge $c = 0$ with $\rho = \gamma + 0 \cdot \alpha = \gamma$ and the challenge $c = 1$ with $\rho' = \gamma + \alpha$. Then it follows that Peggy can compute $\alpha = \rho' - \rho$. In other words, if she doesn't know α she can answer at most one of the challenges, and she must fail at the other challenge. That is, if Peggy does not know α, then she only succeeds at fooling Victor with probability $\frac{1}{2}$. If we repeat the protocol n times, then this probability drops to $1/2^n$.

[1] You may have a look at the following video about these constructions and the impossibility of trisecting an angle, if desired: `https://youtu.be/O1sPvUr0YC0`

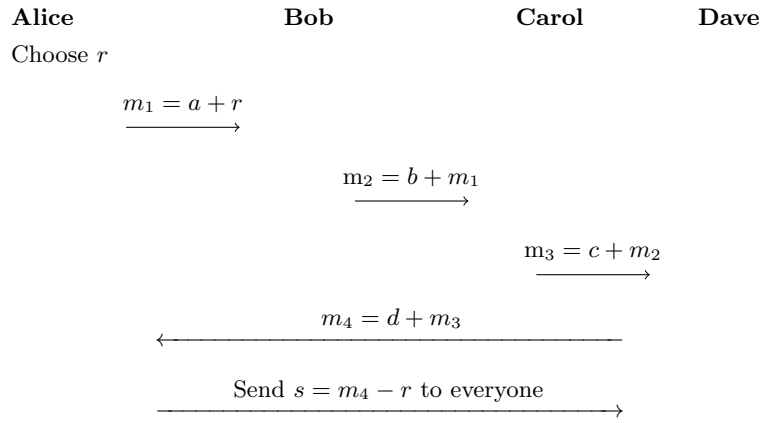
- **Zero-Knowledge.** After running the protocol, Vic has a protocol transcript which consists of the triple $(\tau,\ c,\ \rho)$ from some probability distribution. Vic could, without knowing α, generate an identically distributed fake random transcript: It could choose a random c, generate a random ρ, and construct $\tau = 3\rho - c\beta$. Hence, since the protocol transcript comes from a probability distribution which Vic already knew before running the protocol, Vic does not learn anything new by running the protocol.

  Now, if Vic were a malicious verifier, and it chose the challenge c depending on τ, then the transcript simulation above would not work, and Vic could perhaps learn something about α. The full zero-knowledge property requires that even malicious verifiers cannot learn anything beyond what the protocol proves. The methods to achieve this are out of scope for this course. However, note that since in our protocol the only thing Vic does is choose c randomly, we would be able to achieve the full zero-knowledge property by introducing a mechanism which ensures that c is indeed random. For example, we could force Vic to flip a fair coin in front of Peggy.

2 MPC with Secret Sharing

a) One can easily see that each party after summing locally holds a share of the polynomial $f(x) = f_1(x) + f_2(x)$ (since the degree of the polynomial by summing doesn't change and t points uniquely define a polynomial of degree $t-1$). It follows that $s = f(0) = f_1(0)+f_2(0) = s_1 + s_2$. Hence, if the polynomial is reconstructed and evaluated at point 0, it will give the sum of s_1 and s_2.

b) Dave can just continue the same way as other participants:



c) Alice and Carol can easily compute Bob's salary as follows. Since Carol has $m_2 = b + m_1$ and Alice has m_1 they can compute $b = m_2 - m_1$. The same technique can be used for Dave as well, namely $d = m_4 - m_3$.

d) The main idea is to use secret sharing and its linearity property. Each party shares their salary by using (n, n) Shamir secret sharing, compute locally the sum of each share (of each salary), and in the end reconstruct the sum only.

More concretely, initially, each party has a public ID number and a secret salary. Alice has $(1, s_A)$ (the ID 1 and the salary s_A), while Bob has $(2, s_B)$, Carol has $(3, s_C)$ and Dave has $(4, s_D)$. Each party generates a random polynomial to share their secret salary:

$p_A(x) = s_A + a_1x + a_2x^2 + a_3x^3$	random polynomial of Alice
$p_B(x) = s_B + b_1x + b_2x^2 + b_3x^3$	random polynomial of Bob
$p_C(x) = s_C + c_1x + c_2x^2 + c_3x^3$	random polynomial of Carol
$p_D(x) = s_D + d_1x + d_2x^2 + d_3x^3$	random polynomial of Dave

Then they distribute the shares of each secret and sum all the shares they receive (sum of columns in the following table):

	to Alice	to Bob	to Carol	to Dave
Alice sends	$p_A(1)$	$p_A(2)$	$p_A(3)$	$p_A(4)$
Bob sends	$p_B(1)$	$p_B(2)$	$p_B(3)$	$p_B(4)$
Carol sends	$p_C(1)$	$p_C(2)$	$p_C(3)$	$p_C(4)$
Dave sends	$p_D(1)$	$p_D(2)$	$p_D(3)$	$p_D(4)$
Sum locally	sum_A	sum_B	sum_C	sum_D

Now everyone broadcasts to everyone what the summation of the shares is. For example, Alice sends to everyone sum_A publicly. So do Bob, Carol and Dave. With those values $(sum_A, sum_B, sum_C, sum_D)$ everyone can reconstruct the polynomial (by Lagrange) and evaluate the polynomial at $x = 0$, which is the sum of the salaries.