University of Illinois, Urbana Champaign
CS/ECE 498AC3/4 Applied Cryptography

Instructor: Dakshita Khurana
Scribe: Kuilin Li & Vijayendra Jagtap
Date: October 13, 2020

LECTURE
15

Zero-Knowledge II

In this lecture, we continue the discussion on zero-knowledge proofs with two more examples. In the second half of the last lecture, the concept of zero-knowledge proofs was introduced. The idea behind a zero-knowledge proof is that a verifier exchanges messages with a prover, who knows a secret, in a way that the prover can prove to the verifier that they know the secret, but the entire exchange reveals no information to a third party and could have been fabricated entirely by the verifier, without access to a prover. This ensures that even though the verifier is assured that the prover has the secret, they cannot prove to anyone else that the prover has the secret, which is something the prover may be interested in.

In general, in order to construct a zero-knowledge proof, the prover randomizes and then makes a commitment, and then allows the verifier to choose a limited number of those commitments to open randomly. This ensures that the probability that a dishonest prover is caught is less than 1, and that a third party with a transcript will not be able to be convinced that the prover proved it, if they do not trust the verifier.

We apply this general concept to two examples of zero-knowledge proofs, 3-coloring and Chaum-Pedersen. The 3-coloring protocol allows a prover with a 3-coloring of a graph to prove that the graph is 3-colorable, and the Chaum-Pedersen protocol allows a prover with the keys to prove that an El-Gamal ciphertext is valid ciphertext.

15.1 3-coloring zero-knowledge proof

A 3-coloring of a graph is the assignment of three colors to the vertices of a graph such that every two vertices connected by an edge are different colors. This is an interesting problem because we know from prerequisite courses that, given an arbitrary graph, determining whether it is 3-colorable is an NP-complete problem. If a prover P has a 3-coloring of a graph, and would like to prove to a verifier V that they have a 3-coloring of the graph without revealing the coloring itself, then they can employ this zero-knowledge scheme to verify to just V that they have the 3-coloring and the graph is 3-colorable.

Definition 15.1. A **3-coloring** of a graph (V, E) is defined as a mapping from vertices to colors $C : V \to \{1, 2, 3\}$ such that for any $(v_1, v_2) \in E$, $C(v_1) \neq C(v_2)$. A graph is **3-colorable** if there exists a 3-coloring on the graph.

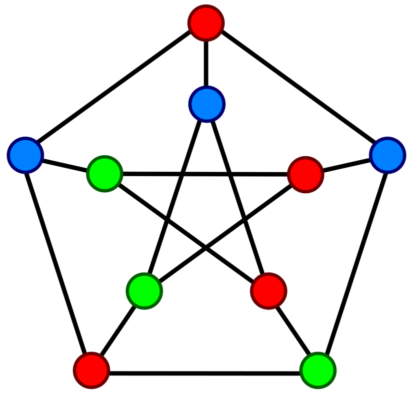

FIGURE 15.1: An example of a 3-coloring of a graph [1]

Figure 15.1 is an example of a 3-coloring of a graph. In the definition, the colors are represented by the numbers $\{1, 2, 3\}$. No two adjacent vertices are the same color, so this graph is properly 3-colored. Note especially that if the colors were permuted, eg. all the red vertices became blue and the blue vertices became red, the resulting graph would still be a valid 3-coloring.

With this definition, we can now describe the 3-coloring protocol:

1. The prover randomly shuffles the colors of the graph to produce a shuffled coloring. Select a random $\pi \in S_3$ and set $C' : V \to \{1, 2, 3\}$ as $C'(v) = \pi(C(v))$ for all $v \in V$.

2. The prover creates a commitment for every vertex that contains its identifier and color, and then sends all these commitments to the verifier. P sends V commit$(v, C'(v))$ for all $v \in V$.

3. The verifier chooses a random edge in the graph $(v_1, v_2) \in E$. V sends P (v_1, v_2).

4. The prover de-commits and reveals the colors of the two vertices of the chosen edge. P sends V $C'(v_1)$ and $C'(v_2)$.

5. The verifier verifies that the revealed vertices are the ones it chose, the colors are not the same, and the commitments were opened correctly.

Recall from the last lecture that a zero-knowledge proof is defined by three properties, completeness, soundness, and zero-knowledge. By proving each of these for the 3-coloring protocol, it can be proved that this protocol is a zero-knowledge proof.

THEOREM 15.2. *The 3-coloring protocol is a zero-knowledge proof.*

Proof. This protocol is complete because if the prover has a real 3-coloring and executes the protocol properly, then the verifier will see two different colors and a successful de-commitment in the final step with probability 1.

The protocol is sound because if the prover does not have a real 3-coloring, then there exists at least one edge (v_1, v_2) such that $C'(v_1) = C'(v_2)$. A verifier therefore has a $\geq \frac{1}{|E|}$

chance of randomly selecting that edge, and if it does, there is no computationally feasible way for the prover to send a proper de-commitment with a different color than what was committed, and if the prover sends back the same two colors, the verifier also finds out. Since the probability of this is nonzero and polynomial, the protocol can be repeated with independent random variables until a reasonable level of assurance is attained. See more notes on this below.

Finally, the protocol is zero-knowledge, because the verifier could, without knowledge of the real 3-coloring, simulate a false prover. In order to simulate a false prover, do the following:

1. The simulator chooses an edge and sets its vertices to two different colors, and the rest to arbitrary colors. First it chooses $(v_1, v_2) \in E$, and then assigns:

$$C(v) = \begin{cases} 1 & \text{if } v = v_1 \\ 2 & \text{else if } v = v_2 \\ 3 & \text{else} \end{cases}$$

2. The simulator shuffles the vertices and sends the commitments identically to how the prover does.
3. The verifier chooses a random edge from the graph.
4. If the random edge does not match the simulator's chosen edge, restart the procedure and transcript.
5. The simulator de-commits and reveals the colors of the two vertices of the chosen edge identically to how the prover does.
6. The verifier verifies that everything is fine.

The probability that the correct edge is chosen and the procedure is not restarted is $\frac{1}{|E|}$, so this simulation scheme runs in polynomial expected time despite brute forcing the random edge by restarting most of the time. The interaction between the simulator and verifier is also identical to the interaction between a real prover and the verifier, because the only difference is that the original coloring $C(v)$ is not a valid coloring, and if it was possible to distinguish the coloring from the commitments alone, then the commitments would be broken. So, this protocol is zero-knowledge.

Therefore, since this protocol is complete, sound, and zero-knowledge, it is a zero-knowledge proof. $\square$

Here's some more analysis on the soundness and number of iterations of the protocol. If the prover does not have a real 3-coloring, then at least one edge will have the same coloring on both vertices. So, the probability that the prover is caught for one iteration of the protocol is $\Pr[\text{protocol fails}|\text{prover is fake}] \geq \frac{1}{|E|}$. So, with n iterations of the protocol, the probability of a false positive is $\leq (1 - \frac{1}{|E|})^n$. If we set $n = |E|^2$, then this value is negligably small, while this scheme still takes polynomial time to run.

15.2 Proofs on encrypted data: Chaum-Pedersen

Recall several lectures ago, when we talked about the El-Gamal encryption scheme, where Alice and Bob generate a shared secret that cannot be determined through the transcript of their communications.

Definition 15.3. The **El-Gamal encryption scheme** is defined as follows. Let p be a large public prime and g be a public generator in the field modulo p. Alice first generates a random a and sends g^a to Bob. Bob generates a random b and sends g^b to Alice, and then computes $(g^a)^b$. Then, Alice computes $(g^b)^a$ and now they have a shared secret g^{ab} while the transcript is (g^a, g^b).

Note that the transcript (g^a, g^b) is not enough to find g^{ab} in polynomial time, assuming the hardness of the discrete log. The Chaum-Pedersen protocol can be used to prove that a ciphertext encrypted using El-Gamal is valid, without exposing the plaintext.

For simplicity, we will work with an equivalent form of the problem. For honest provers, let $u = g^a$, $v = g^b$, and $w = g^{ab}$, and these three variables are public. What the prover would like to prove to the verifier now becomes proving that there exists a and b such that u, v, and w are generated from them. The prover has a and b, but does not want to reveal any information about them in the process of this proof.

The Chaum-Pederson protocol is a zero-knowledge protocol to accomplish this. It follows the same model as 3-coloring, where first the prover makes a commitment, and then the prover selects a random test case, with which the commitment is then validated. A key difference is that in this scheme, technically the commitment is not de-committed, but that is fine because arithmetic on the commitment values v' and w', after the prover replies to the challenge c with a response e, can allow the verifier to verify the validity of the test case anyways.

Definition 15.4. The **Chaum-Pederson protocol** is defined as follows, for $u = g^a$, $v = g^b$, and $w = g^{ab}$:

1. The prover generates a random d and computes $v' = g^d$ and $w' = g^{ad}$, and then sends (v', w') to the verifier.
2. The verifier generates a random c and sends it to the prover.
3. The prover calculates $e = d + bc$ and sends it to the verifier.
4. The verifier checks that $g^e = v'v^c$ and $u^e = w'w^c$.

Following the same pattern as the 3-color protocol, we can again prove that this protocol is a valid zero-sum proof protocol by proving its defining properties. Completeness and soundness are demonstrated by definition chasing, and zero-knowledge is demonstrated by constructing a simulator that produces a conforming transcript in polynomial time without knowledge of a or b, which demonstrates that a transcript would not be convincing to a third party.

Theorem 15.5. *The Chaum-Pederson protocol is a zero-knowledge proof.*

Proof. The protocol is complete because if the prover is honest, then $g^e = g^{d+bc}$ and $v'v^e = g^d(g^b)^c = g^{d+bc}$ so the first check passes, and $u^e = (g^a)^e = g^{ad+abc}$ and $w'w^e = g^{ad}(g^{ab})^c =$

g^{ad+abc}, so all the checks will pass.

The protocol is sound. If the relationship between u, v and w is incorrect, we can define a using the discrete log of $u = g^a$ and b using the discrete log of $v = g^b$, and then we see $w \neq g^{ab}$ because otherwise the relationship would be correct. Next, the e sent from the prover is either $d + bc$ or not $d + bc$. If $e = d + bc$, then $u^e = (g^a)^e = g^{ad+abc}$ is still true, but $w'w^e \neq g^{ad}(g^{ab})^c = g^{ad+abc}$ so $u^e \neq w'w$, so the second check would fail. If $e \neq d + bc$, then $v'v^c = g^d(g^b)^c = g^{d+bc}$ but $g^e \neq g^{d+bc}$, so the first check would fail. Thus, if what is to be proved is incorrect, then at least one of the checks will fail.

Finally, the proof is zero-knowledge. The verifier can construct a simulator that simulates a prover, and then do the following:

1. The simulator generates a random e first, and picks a random c.
2. The simulator makes v' and w' so that $d = e - bc$. It can't directly generate d, because it doesn't know b, but it can generate $v' = g^d = g^{e-bc} = g^e(v^c)^{-1}$ and $w' = g^{ae-abc} = u^e(w^c)^{-1}$ using only public information.
3. The verifier generates a random c' and sends it to the prover.
4. If $c' \neq c$, then restart.
5. The simulator sends the e it generated at the beginning.
6. The verifier does the two checks.

Note that by construction, we do have $e = d + bc$. So, both checks will pass. The probability that $c' \neq c$ is $\frac{1}{p}$ because $c \in \mathbb{Z}_p$, so retrying that until a hit is achieved is still polynomial time with probability approaching 1. Therefore, a simulator can be used to produce such a transcript quickly without knowledge of the secrets, so this protocol is zero-knowledge.

Since the protocol is complete, sound, and zero-knowledge, it is a zero-knowledge proof.

$\square$

Acknowledgement

These scribe notes were prepared by editing a light modification of the template designed by Alexander Sherstov.

References

[1] W. Commons. File:Petersen graph 3-coloring.svg — Wikimedia Commons, the free media repository, 2020. [Online; accessed 17-October-2020].