

Modern Cryptology (236506) – Zero-Knowledge Tutorial

1 Bit Commitment

Imagine a talk-show, where a magician writes down on a note the winning numbers of the next lottery poll, and deposits the note inside a closed envelope and inside a safe which is locked with a key. The safe is kept by a trusted third party, so that the magician does not have access to it. In the next talk-show after the lottery poll the safe is opened (using the key), and from inside of the envelope the winning numbers are shown to the applauding audience.

Cryptographically, we require the magician to commit on values, that remain secret until they are opened, and cannot be opened as other values. We can see the commitment as a game between a sender and a receiver. The bit commitment is composed out of two phases:

1. Commit — In this phase the sender commits to the value S , by depositing the value C with the receiver. The input to this phase are the secret value S , which the sender commits to, and the secret key r , which is later used to open the commitment (equivalent to the key of the safe). The output of the commit phase is the commitment $C = C(S, r)$ that the sender gives to the receiver, and from which the receiver cannot recover S . Note that the value $C(S, r)$ corresponds to the sealed safe.
2. Reveal — In this phase the sender sends its claimed value and key (S', r') to the receiver, and the receiver checks whether $C(S', r') = C$. The receiver outputs a true/false indication to whether this is a correct opening of the commitment.

We require two important properties of a bit commitment scheme:

1. Secrecy — The receiver must not be able to deduce S (or any information about S) from the value $C(S, r)$.
2. Unambiguity — The sender must not be able to provide $(S', r') \neq (S, r)$ such that $C(S, r) = C(S', r')$, i.e., reveal a value different than the one it committed to.

Note that the secrecy property protects the sender from the receiver, while the unambiguity protects the receiver from the sender.

It suffices to show how to commit on a single bit. In case we need to commit on several bits, we repeat the scheme for each bit of the value.

1.1 A Perfect Binding, Computational Secrecy Scheme

Let p be a large prime, and g a generator of Z_p^* , where p and g are known and agreed by both parties. In order to commit on a value $S \in \{0, 1\}$, the sender chooses a random $0 \leq r \leq p - 2$ such that the parity of r (in its binary representation) is S . The sender computes $C \equiv g^r \pmod{p}$, and sends C to the receiver. In the reveal phase the sender sends (S, r) to the receiver. When the receiver receives (S', r') , he verifies that $g^{r'} \equiv C \pmod{p}$.

The sender cannot reveal a value $r' \neq r$, because $g^{r'} \equiv g^r \Rightarrow r' \equiv r$ (g is a generator in Z_p^*). Therefore, the property of unambiguity holds regardless of the computation power available to the sender. We refer to this case as *perfect binding*. However, if the receiver invests enough computing power, he can find the discrete logarithm of C to base g modulo p , i.e., it can recover r , and therefore recover S (e.g., r can be found by trying all possible values r' and testing whether $C \equiv g^{r'}$). We refer to this case as *computational secrecy*.

1.2 A Computational Binding, Perfect Secrecy Scheme

Let p be a large prime, and q another prime such that $q|p - 1$. Therefore, there are elements in Z_p^* of order q . Denote the cyclic group of all elements of order q (and the element 1) by G_q . G_q is cyclic, i.e., all elements of G_q (except for 1) are generators. Let h and g be generators of G_q . The parameters of the scheme are agreed between the parties before the protocol, i.e., p , q , h , and g are public. Since g is a generator we can write $h \equiv g^l \pmod{p}$ for some l . l is the discrete log of h to base g . We require that l is unknown (for example h and g are chosen by a trusted center).

To commit on a value $S \in Z_q$, the sender chooses $r \in Z_q$ at random, and computes $C \equiv g^S h^r \pmod{p}$, and sends the commitment C to the receiver. In the reveal phase the sender reveals (S, r) to the receiver. After the receiver receives (S', r') , he verifies that $C(S, r) \equiv g^{S'} h^{r'} \pmod{p}$. Note that the secrecy in this case is perfect, because for every S that is chosen, every C can be selected, and there is a 1-1 correspondence between the possible C values and the possible r values. In other words r perfectly masks the value of S . On the other hand, the binding is computational. We now show that the binding is equivalent to the discrete log problem. Assume that the sender can reveal a value $(S', r') \neq (S, r)$ such that $g^{S'} h^{r'} \equiv g^S h^r \pmod{p} \Rightarrow g^{S'} (g^l)^{r'} \equiv g^S (g^l)^r \pmod{p} \Rightarrow S' + lr' \equiv S + lr \pmod{q} \Rightarrow l \equiv \frac{S - S'}{r' - r} \pmod{q}$. Therefore, if the sender can reveal a different value, he has found the discrete logarithm l of h to base g .

Note that it is not possible to gain both perfect secrecy and perfect binding: Assume we have perfect binding, it means that W.L.G. the value C can be the result of $S = 1$ with some r , but not the result of $S = 0$ (otherwise, the sender would be able to reveal the other value). Therefore, the receiver can try all the possible (S, r) values, and test which one leads to C , which implies a computational secrecy. On the other hand, if we have perfect secrecy, it means that a specific value

C can be the result of both $S = 1$, and $S = 0$, therefore, the sender can try all possibilities of r' and find one that for the complementing value of S results in the same C , which implies computational binding.

2 Zero Knowledge Proof for G3C

In this section we give a Zero Knowledge protocol for *Graph 3-Colorability* — G3C. Let $G = (V, E)$ be a graph, $\rho : V \mapsto \{1, 2, 3\}$ a coloring of the graph. The graph G is said to be in G3C, if there exists ρ such that for every $(u_1, u_2) \in E$, $\rho(u_1) \neq \rho(u_2)$. G3C is NP-Complete, so by showing a Zero-Knowledge protocol for G3C, we actually show that every language in NP has a ZK protocol (Use the reduction of that language to G3C, and execute the protocol for G3C).

The idea of the protocol: The prover claims that a graph $G = (V, E)$ is 3-colorable, and he has a 3-coloring ρ of the graph. The prover wishes to convince a verifier that the graph is 3-colorable, without revealing anything about ρ . The idea is that the prover chooses a random permutation on the colors of ρ (note that a random permutation of a 3-coloring of a graph is also a 3-coloring of a graph). Then the prover commits on the coloring, and lets the verifier challenge him. The verifier chooses an edge $(u_j, u_k) \in E$ of the graph, and requests the prover to reveal the color of the two vertices u_j, u_k of the edge. The prover reveals the commitment on these vertices, and the verifier checks that the colors are different. The protocol repeats many times. If in one of the iterations the vertices are of the same color, the verifier aborts and claims that the prover is cheating.

Given a graph $G = (V, E)$, $n = |V|$, we denote the 3-coloring of the graph which is available to the prover by $\rho(u)$ for a vertex u . The protocol is as follows:

1. The prover P Chooses a random permutation $\pi : \{1, 2, 3\} \mapsto \{1, 2, 3\}$.
 P creates commitments $C_i \triangleq C(\pi(\rho(u_i)), r_i)$, $i = 1, \dots, n$, and sends C_1, \dots, C_n to the verifier V .
2. V chooses $(u_j, u_k) \in E$ at random, and sends (u_j, u_k) to P .
3. P sends $(\pi(\rho(u_j)), r_j), (\pi(\rho(u_k)), r_k)$ to V .
4. V checks that $C_j = C(\pi(\rho(u_j)), r_j), C_k = C(\pi(\rho(u_k)), r_k)$, and accepts/rejects accordingly.

The protocol is repeated many times to increase the verifier's confidence.

We first need to show that this is indeed an interactive proof. We show completeness and soundness. Completeness: For every graph $G \in G3C$ the prover P can always convince V , since every two vertices with a common edge are always of different colors. Soundness: we show that for every $G \notin G3C$, and for every prover P^* , the verifier accepts with low probability: Let $G \notin G3C$. Then, the commitment of P^* must have an edge $(u_j, u_k) \in E$ such that the committed values on u_j and on u_k are equal (i.e., the two vertices are of the same color). Therefore, the verifier rejects with probability at least $1/|E|$. He thus accepts with probability of at most $1 - 1/|E|$. If we repeat the protocol $|E|$ iterations, and the graph is large enough, then the verifier accepts with probability at most $(1 - 1/|E|)^{|E|} \approx 1/e < 0.5$, i.e., the verifier rejects in most cases. By repeating $|E|^2$ iterations, the probability that the verifier accepts is bounded by $1/e^{|E|}$. Note that we must use a commitment

scheme with perfect binding, otherwise if the binding is computational, the soundness argument fails: a prover with infinite computation power can open his commitment in a different way than it committed, thus causing the verifier to accept $G \notin G3C$. Thus we must use perfect binding and computational secrecy.

We now show that the protocol is computational zero knowledge. The philosophy is that for every verifier, no matter how much he tries to cheat, we show an algorithm (a *simulator*) that the verifier can use to create “at home”, without interacting with the prover, a *transcript* of the conversation with the same statistical distribution as this verifier would get while interacting with the prover. Since the verifier can create a similar transcript “at home” he learns nothing from the interaction with the prover (except for the fact that the claim holds). We prove that this protocol is computational zero knowledge, where the distribution of the transcript of the simulator and the transcript of a real interaction between the verifier and the prover are different, and that these distributions are *computationally indistinguishable*.

The simulator for verifier V^* , and graph $G = (V, E)$:

1. Choose a random color $S_i \in \{1, 2, 3\}$ for every vertex u_i in the graph, and create a commitment $C_i = C(S_i, r_i)$. “Send” C_1, \dots, C_n to the verifier V^* .
2. V^* returns an edge $(u_j, u_k) \in E$. If the colors of u_j and u_k are the same (i.e., $S_j = S_k$), restore the state of V^* to the state it had before step 1, and restart the iteration again from Step 1. Otherwise (if the colors are different) continue to Step 3.
3. Append the commitments C_1, \dots, C_n , the chosen edge (u_j, u_k) , and the openings of the commitments $(S_j, r_j), (S_k, r_k)$, to the transcript.

The simulator is repeated the same number of iterations as the original protocol.

We analyze the correctness of the simulator as follows. We first claim that the simulator stops after a polynomial number of iterations (amortized). Each edge $(u_j, u_k) \in E$ that V^* chooses in the second step, has a probability of $1/3$ to have the same color at both ends (due to the randomness in the first step), and thus, the simulator fails in Step 2 with probability $1/3$. Therefore, we expect the simulator to run $(1 - 1/3)^{-1} = 1.5$ times on average for each time it passes Step 2 successfully, and therefore completes its run for the relevant iteration.

Note that since we use computational secrecy in the bit commitment scheme, we rely on the fact that V^* has a polynomial running time, and therefore, cannot see through the commitments, i.e., the secrecy holds. (Had it been otherwise, since the coloring is random V^* could have made the simulator consistently fail by choosing the edge that violates the 3-colorability.)

We claim that the distribution of the transcript generated by the simulator and the transcript of a real interaction are computationally indistinguishable. With an unlimited computational power the (computationally secret) commitments can be opened, and we see that the coloring of the simulator in any step is (probably) not a 3-coloring, while the real prover’s coloring is 3-colorable. However, when dealing with computational zero knowledge these distributions should be computationally indistinguishable. Assuming that the commitment scheme cannot be broken using a polynomial-time algorithm, the commitments are computationally indistinguishable, and the commitment opening reveals two different random colors in both cases. This is not a formal proof, but

it should give the general idea. In a formal proof we should show that if someone can distinguish the transcript generated by the simulator from the transcript of the protocol, then he can break the secrecy of the commitment scheme.