Bit Commitment

1. Introduction

The Problem: Alice wants to commit to a prediction (i.e., a bit or series of bits) but does not want to reveal her prediction until sometime later. Bob, on the other hand, wants to make sure that Alice cannot change her mind after she has committed to her prediction. The assumption here is that there is no third party that is trusted by both parties, otherwise the problem would be trivial.

The first solution that comes to mind is that Alice can encrypt the message and send it to Bob, and when it's ready to be revealed she can just decrypt it. The problem here is that if the message is small (maybe just a few bits), finding two keys which will encrypt the encrypted text to two different plain texts is too easy. As the name suggests the trick in these protocols is being able to handle messages which might be only 1 bit.

2. Bit Commitment Using Symmetric Cryptography

Bob first sends Alice a random number. Alice then concatenates her message with that random number and encrypts it (with a random key), then she sends the result to Bob. When its time to reveal the message she can send Bob the key, and Bob can check the random number he sent to verify the message. The random number here makes it harder for Alice to generate two keys.

3. Bit Commitment Using One-Way Functions

1. Alice generates 2 random strings $R_1$ and $R_2$. She then computes the one-way function $H$ on $(R_1, R_2, b)$, where $b$ is her message. She then sends the result along with one of the random strings, $R_1$.
2. When the time comes, Alice sends Bob the original message $(R_1, R_2, b)$.
3. Bob checks if $R_1$ is correct and then computes the $H$ of the message and checks the result with Alice's version. If both of them match, the message is verified.

The benefit of this protocol over the previous one is that Bob does not have to send any messages.

4. Bit Commitment Using Random Number Generators

1. Bob sends Alice a random string $s$
2. Alice generates a random seed $x$, and then sends Bob $f(s,x)$, where $f$ is defined as
   1. The output of the generator, if Bob's bit is 0
   2. The XOR of the output of the generator and her bit (from her message) if Bob's bit is 1
3. When it's time to reveal the message, Alice sends Bob her random seed, Bob can confirm that Alice was acting fair.

**Fair Coin Flips**

1. **Introduction**

The problem this time is to find a way for Alice and Bob to flip a coin online (or on the phone) A protocol using Bit Commitment is as follows:

1. Alice commits to a random bit, using any of the bit-commitment schemes.
2. Bob tries to guess the bit.
3. Alice reveals the bit to Bob. Bob wins the flip if he correctly guessed the bit.

2. **Coin Flipping Using One Way Functions**

1. Alice chooses a random number, $r$ and computes $f(r)$, where $f$ is a one-way function.
2. Alice sends Bob the result.
3. Bob guesses whether $r$ is even or odd and sends her guess to Alice.
4. Alice sends Bob the random number so that Bob can check $f(r)$
5. Bob wins if he guessed correctly.

3. **Coin Flipping Using Public Key Cryptography**

Actually it is not necessary that we use public key cryptography here, but the algorithm must be commutative. (RSA is with identical moduli)

1. Alice and Bob generate a public/private key pair
2. Alice encrypts two messages with her public key. The messages represent tails and heads and have a random string attached to them. She then sends both of the messages in a random order.
3. Bob chooses one of the messages (randomly) and encrypts it using his public key and sends it back to Alice.
4. Alice cannot read the message. She just decrypts it using her private key and sends it back to Bob.
5. Bob decrypts the message and sends it back. Both can see the result of the flip now.
6. They exchange their private keys to make sure that no one cheated.

4. **Applications**

A real application for this protocol is session-key generation. Coin-flipping protocols allow Alice and Bob to generate a random session key such that neither can influence what the session key will be. And assuming that Alice and Bob encrypt their exchanges, this key generation is secure from eavesdropping as well.

*References:* Applied Cryptography, 2nd Edition, B. Schneier