CS181A Notes #7 Miscellaneous Protocols

We describe several simple but interesting cryptographic protocols.

Coin Tossing Suppose Alice and Bob would like to agree on a common bit value b. They would like to come up with a fair protocol where one party cannot influence the outcome (which is the value of the common bit b) of the protocol more than the other party. If a trusted third party exists, then the following simple protocol suffices: Alice and Bob send their bits to the third party, say b_A and b_B, and then the third party will compute the XOR of the bits $b = b_A \oplus b_B$, and announce this as the value of the common bit.

In the absence of a trusted third party, the following protocol due to Blum and Rabin is notably of interest.

1. Alice selects a Blum integer $N = pq$ and sends N to Bob (keeping the primes p and q secret).
2. Bob selects a random $x \in \mathbb{Z}_N^\star$, computes $z \equiv x^2 \pmod{N}$, and sends z to Alice (keeping x secret).
3. Alice computes the four square roots of z, say $\pm x$ and $\pm y$ modulo N, where $x^2 \equiv y^2 \equiv z \pmod{N}$ but $x \not\equiv \pm y \pmod{N}$. Alice then guesses Bob's square root as $\pm x$ or $\pm y$. She sends her guess to Bob.
4. Bob announces to Alice if her guess is correct (in which case, Alice decides the bit), or not (in which case, Bob decides the bit).

If Bob declares that Alice's guess is incorrect, he can prove this fact by sending Alice the primes p and q. Bob can obtained this by computing $\gcd(x \pm y, N)$. But it is harder for Alice to ascertain the other outcome (when Bob declared Alice is correct). This can be fixed using the idea of *bit commitment*.

Zero Knowledge Proofs Suppose Alice (the Prover) wants to convince Bob (the Verifier) that a certain statement is true. We allow an *interactive* protocol where Alice and Bob may exchange messages (queries and responses) based on the specification of the protocol. At the end of the protocol, Bob must decide whether to accept or to reject that the statement is true. Both Alice and Bob are allowed to make probabilistic moves or decisions (they have sources of random bits at their disposal). The probabilistic interactive protocol is required to satisfy the following properties:

1. (Completeness) If the statement is true (Alice is a honest Prover), then there is a strategy for Alice where Bob accepts with probability at least $1 - \epsilon$.

2. (Soundness) If the statement is not true (Alice is a dishonest Prover), then for any strategy that Alice employs Bob rejects with probability at least $1 - \epsilon$.

3. (Zero knowledge) At the end of the protocol, Bob learns nothing other than whether the statement is true or not.

The idea of probabilistic interactive proofs and zero-knowledge is due to Goldwasser, Micali and Rackoff (1985).

As an example of a zero-knowledge protocol, we describe a protocol for Quadratic Residuosity due to Fiat and Shamir. Here, Alice (the Prover) would like to prove to Bob (the Verifier) that an integer $x \in \mathbb{Z}_N^\star$ is a quadratic residue modulo N. A simple protocol (that is not zero-knowledge) is for Alice to send one of the square roots of x to Bob. But, then Bob learns *more* than just the fact that x is a quadratic residue modulo N.

ZKP for Quadratic Residuosity
The common inputs to Alice and Bob are N and $x \in \mathbb{Z}_N^\star$.
The statement that Alice is claiming is $\boxed{x \in QR_N}$.
Suppose that y is a square root of x modulo N, that is, $y^2 \equiv x \pmod N$ (we assume for now that Alice is honest).

1. Alice selects a random u and computes $v \equiv u^2 \pmod N$. She sends v to Bob (while keeping u secret).
2. Bob selects a random bit $b \in \{0, 1\}$. He sends b to Alice.
3. Alice computes $z = uy^b \pmod N$. She sends z to Bob.
4. Bob accepts Alice's claim (that $x \in QR_N$) iff $z^2 = vx^b \pmod N$.

We omit the correctness proof of the above protocol from this draft. A significant improvement to the above protocol was given by Feige, Fiat and Shamir (1990).

Bit Commitment Alice wants to *commit* a bit to Bob which she will only reveal at a future time. In the meantime, Alice should not be able to change the value of that bit and Bob should not be able to learn the value of this bit. Bob will learn the value of this bit only when Alice reveals her bit in a *decommitment* phase.

We can derive a bit commitment protocol from a simple application of the Goldwasser-Micali probabilistic encryption function.

1. Alice selects a Blum integer $N = pq$ and an integer y where $\left(\frac{y}{p}\right) = \left(\frac{y}{q}\right) = -1$. She sends N and y to Bob.

2. Alice commits a bit $b \in \{0, 1\}$ by selecting a random $x \in \mathbb{Z}_N^\star$ and sending $z \equiv x^2 y^b \pmod{N}$ to Bob.

3. Alice decommits the bit by sending p and q to Bob.

Note that it is impossible for Alice to change her bit since she cannot turn a quadratic residue into a quadratic non-residue (and vice versa). Also, Bob cannot learn the value of the bit by the assumed hardness of the Quadratic Residuosity problem for Blum integers.

Oblivious Transfer Alice sends m items to Bob and Bob can only receive one and only one of these items. Moreover, Alice will not know the item which Bob obtained. This is the so-called 1-in-m *oblivious transfer* (OT) protocol. We will show that this protocol is immensenly useful for secure two-party computation of the MAX function. In what follows, we describe a simple realization of 1-in-m OT.

1. Suppose that Alice has m items $x_1, \ldots, x_m$ to send to Bob.

2. Alice selects m random encryption functions $f_1, \ldots, f_m$ (say Goldwasser-Micali). Suppose $f_i : \{0, 1\}^k \to \{0, 1\}^k$. She sends these m functions to Bob.

3. Bob selects a random $r \in \{0, 1\}^k$ and a function f_i (from the m sent). He computes $y = f_i(r)$ and sends this to Alice.

4. Alice computes $c_i = f_i^{-1}(y) \oplus x_i$. She sends $c_1, \ldots, c_m$ to Bob.

5. Bob recovers $x_i = c_i \oplus r$. Alice does not know the identity of i.

Further properties of oblivious transfer in connection with secure multiparty computation were studied by Kilian (1988).

Yao's Millionaires problem As an application of the Oblivious Transfer protocol, consider a scenario where Alice and Bob each have secret values v_A and v_B, respectively. They would like to compute $\max(v_A, v_B)$ together but do not want to reveal to each other their respective secret values. Suppose that they agree to partition the domain of their values to $I_1, \ldots, I_m$. Alice constructs a Boolean array A of size m where $A[i] = 1$ iff $v_A \geq I_i$. Using 1-in-m OT, she sends these m Boolean values to Bob. Bob reveals the value of I_{v_B}. The latter value is 1 iff $v_A \geq v_B$. Bob announces this result to Alice.

Secret Sharing Alice would like to share a secret s with N of her friends. She would like the property that to recover s, at least k of her friends must be involved. This is called a (k, N)-threshold secret sharing. Here, we describe Shamir's simple idea of using polynomials to achieve these requirements. We take a polynomial f of degree $k - 1$ (over some field) and note that we need at least k points to recover such a polynomial. Thus, the secret can be the polynomial itself (its coefficients, say) and the distributed pieces of the secret are the evaluation points of f at various points. Recovery of the polynomial can be done using Lagrange interpolation.