1 Week 10 Workshop:

1.1 Byzantine Agreement

Byzantine Agreement requires $t+1$ rounds and a total of at least $3t+1$ nodes to tolerate t traitors. Assume that the general begins by giving the order, but then behaves like an ordinary participant for the rest of the protocol.

1. In class we covered the Exponential Information Gathering (EIGByz) Algorithm, for Byzantine failures. Think of an example with 2 traitors and 4 honest participants in which the honest participants do not agree, even though they follow EIGByz correctly.

   Ans: One of the traitors is the general, who divides the 4 honest participants into 2 pairs. One pair are told to attack, the other pair to retreat. Then the 2 traitors (the general and one other) lie to the honest participants about what the other pair of honest participants have told them. (There could be other good solutions too.)

2. We also covered a simpler version of Exponential Information Gathering appropriate for Authenticated Byzantine Agreement (slide 18). Give an example with two traitors in which the honest participants do not agree after t rounds, even if they follow this simplified EIG algorithm exactly. Explain why all honest participants must agree after $t+1$ rounds.

   Ans: See sketch in notes on bottom of p. 19.

1.2 Message Authentication and Hashing

1. **The "birthday paradox"** Let H be a hash function that outputs a k-bit digest. (Assume for simplicity that it's a single-block hash function with fixed-length inputs that are also k bits. It doesn't really matter though.) You are trying to generate collisions by guessing. Show that after guessing about $2^{k/2}$ different input values, there is a decent probability of having found at least one pair of different values A and B with $H(A) = H(B)$.

   (Hint: For small $x, e^{-x} \approx 1-x$. Bigger hint: see Paar and Pelzl's texbook, Ch 11.)

   This is an important result, and the main reason that recommended hash lengths tend to be double the recommended symmetric-encryption key lengths.

   Ans: See Paar and Pelzl, pp $300 - -302$.

1.3 Applications

1. **Randomised partial checking** Suppose two adjacent mix servers A and B shuffle and rerandomise some encrypted values, and an independent (trustworthy) party verifies their correctness using randomised partial checking. Each ciphertext of A's output, which is also B's input, is randomly selected to have either its source or

its destination revealed, each with probability 1/2. (Suppose also that each proves that no two outputs derive from the same input, and no two inputs are linked to the same output—see the paper by Juels, Catalano and Jakobsson if interested.)

(a) If A substitutes one input, what is the probability that it gets caught?
Ans: $1/2$.

(b) If B substitutes k outputs, what is the probability that it gets caught?
Ans: $1 - 1/2^k$.

(c) If A and B collude together and substitute k inputs and k outputs, can they do so stupidly so they are guaranteed to get caught?
Ans: Yes, if they manage to cheat on both the source and the destination of one of the middle ciphertexts.