Advanced Cryptography

GDPR

- A practical (Legal) motivation.
- EU's General Data Protection Regulation
- Data Processors must obtain consent from data subjects before processing their data!
- They should inform how data will be used
- Data subjects have rights to:
  - erasure, access, rectify, restrict processing, portability.
- Data subjects have special personal data:
  - race/ethnicity, political opinion ....

Schrems II

- Decision by the Court of Justice of the European Union in 2020
- European companies acting as data controllers or processors use cloud services

Multi Party

- Parties $P_1 \ldots P_n$ want to compute $f(x_1, \ldots, x_n)$
- Follow protocol sending messages over network until obtain $y = f(x_1, \ldots, x_n)$
- # of sent messages might depend on the number of parties!

Adversarial Models

- Honest parties: always follow protocol
- Formally there is one Adversary who corrupts parties
- Corrupted parties are controlled by the adversary.

Types

Honest but Curious: Don't deviate from protocol but try to break its security by observing

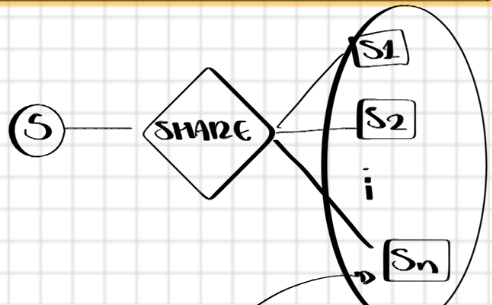
Malicious: Deviate from protocol in order to break its security

Rational & covert: Deviate if it is profitable, cost of breaking offsets gettin' caught!

Static adversary: corrupts all parties before protocol execution starts.

Adaptive adversary: corrupts all parties at any point before or after execution started

t-out-of-n Threshold Secret Sharing



subset with enough info though reconstructs s

not enough infos to original secret unless enough shares are known!

n-out-of-n

- An n-out-of-n scheme can be obtained on a field F_q by simple addition.
  1. Sample n-1 random shares $s_1, ..., s_{n-1}$ from F_q
  2. Compute $s_n = s - (s_1 + ... + s_{n-1})$ where s = secret
  3. Final shares vector is $(s_1, ..., s_n)$
- Final s can only be obtained if all shares $(s_1, ..., s_n)$ are known.
- One missing s acts as one-time-pad.

Hash Based Commitments

- Use a cryptographically secure Hash Function $H: \{0,1\}^* \rightarrow \{0,1\}^k$
- Computing a commitment $Com(m,r)$:
  - Alice samples a random k-bit string r & computes $c = Com(m,r) = H(r|m)$
  - Intuitively it is hard to invert a hash or find preimage.
  - When Alice sends c to Bob, he cannot find $r|m$
  - Using randomness r is important, otherwise m is predictable by bruteforcing!
- Opening a Commitment:
  - Alice sends $r|m$ to Bob who tests if $c = H(r|m)$
  - It is very hard to find a collision $r'|m' \neq r|m$ such that $H(r|m) = H(r'|m')$
  - Alice cannot lie

Coin tossing

- Alice & Bob want to generate a random bit such that none of them can predict it.
- Coin tossing Protocol by Blum:
  1. Alice samples random bit & sends commitment $Com(a,r)$ to Bob
  2. Bob does same & sends to Alice
  3. Alice sends (a,r) to Bob, who checks that it is a valid opening for $Com(a,r)$
     Both compute a XOR b

- BOB DOES NOT KNOW ALICE'S BIT a SO CANNOT INFLUENCE OUTPUT
- ALICE CANNOT CHANGE HER BIT a AFTER SEEING BOB'S bit b. SO CAN'T INFLUENCE