Alice and Bob’s Revenge?

If there is a protocol

If there is a protocol then there must be a shortest protocol

1. Alice → Bob : ??
2. Bob → Alice: ??
   ....
3. ?? → ?? : ??
4. ?? → ?? : ??

Modelling and Analysing of Security Protocol: Lecture 13

What you can’t do with protocols ..... and how to do it anyway.

Tom Chothia
CWI

The Rest of the Course

• 2nd, 9th, 16th, 23rd and 30th Nov
  Student presentations.
• 2nd:
  – Antanas Kaziliunas
  – Liang Wang
  – Kristian Rietveld
  – Renuka Autar

Final deadline: e-mail about your presentation by Wednesday

Today

• What you can’t do with protocol: global consensus
• Activities that require global consensus
• Global consensus using probability or Trusted Third Party.
BREAK
• Some commonly used protocol
• Extracting a protocol from a RFC
What’s Going Wrong?

• Using BAN logic:
  A \equiv B \rightarrow A will attack
But also:
  B \equiv A \rightarrow B will attack
Therefore:
  A \equiv B \equiv A will attack
  \Rightarrow A will attack

What’s Going Wrong?

Therefore:
  A \equiv B \equiv A will attack
  \Rightarrow A \equiv B \equiv A will attack
  \Rightarrow A \equiv B will attack
  \Rightarrow A will attack

and so on and so on ....

Alice and Bob need global consensus, i.e., for every n:

• A \equiv B \equiv ... n \equiv B \equiv X
  And:
  B \equiv A \equiv ... n \equiv A \equiv X

Each message can only add a finite “amount of belief”

No true global consensus unless you can be 100%
sure that your message will be delivered in a fix
amount of time.

Problems that Require Global Consensus

• Atomic Commitment
• Leadership Election
• Fair Exchange
• Synchronous communication from
  asynchronous channels.

How To Do It Anyway

• We can approximate global consensus
  by:
    – Using probability
    – Using a Trusted Third Party
    – Using unique IDs for each process

Or we can just not use it.

Synchronous Channel from
Asynchronous Channels

1. A \rightarrow B : Message
2. B \rightarrow A : ACK

Not global consensus:
  – B can claim it did get the message when it did.
  – A might not get the ACK message.
But still useful
Two Phase Commit

- There are many participants and one administrator:

  Participants → Admin : Prepared
  Admin → Participants : Commit

Leadership Election

- What happens if the Admin fails?
- Easy if the nodes have unique IDs.
- Otherwise hard to break symmetry.
- Have to use probability.

Fair Exchange using a TTP

1. $A \rightarrow S : \{B, M_1, Sign_A(M_2)\}_{K_{as}}$
2. $B \rightarrow S : \{A, M_2, Sign_B(M_1)\}_{K_{bs}}$
3. $S \rightarrow A/B : Sign_A(M_2), Sign_B(M_1)$

Rabin’s F.E. Protocol

- $A,B$ and $S$ agree on "$n$" such that they want to succeed with prob. $(n-1)/n$

  $A \rightarrow B : Sign_A(M_a$ if $S$ says 1)
  $B \rightarrow A : Sign_B(M_b$ if $S$ says $<= 2$)
  $A \rightarrow B : Sign_A(M_a$ if $S$ says $<= 3$)
  ....
  $B \rightarrow A : Sign_B(M_b$ if $S$ says $<= n-1$)
  $A \rightarrow B : Sign_A(M_a$ if $S$ says $<= n$)
  $B \rightarrow A : Sign_B(M_b$ if $S$ says $<= n$)
  $S \rightarrow A,B : Sign_S($random number between 1 & $n$)

1 of 2 Oblivious Transfer

- Probabilistic Fair Exchange without a trusted third party often use Oblivious Transfer.
- 1 of 2 Oblivious Transfer lets Alice send one of two messages to Bob in such a way that Alice doesn’t know which message was sent!
1 of 2 Oblivious Transfer

- Alice wishes to send M1 or M2, she picks two numbers “n” and “m”:
  1. A → B : n, m
  2. B → A : EA(k) + ?  (k random, ? = n or m)
  3. A → B : Dec( EA(k) + ? - n ) + M1
     , Dec( EA(k) + ? - m ) + M2

Fair Exchange using Oblivious Transfer

- Fair Exchange assumes that
  - Alice and Bob can recognize the exchanged data
  - The data is uniformly reconstructable.
- Alice and Bob exchange half their data using OT then reveal all the data bit by bit.

Summary

- Global Consensus is not possible but can be approximated using
  - A trusted third party
  - Probability.

Today

- What you can’t do with protocol: global consensus
- Activities that require global consensus
- Global consensus using probability or Trusted Third Party.
  BREAK
- Some commonly used protocol
- Extracting a protocol from a RFC