# Cryptography with Everyday Objects

## Mucking about with cards and stuff

James Heather[1] Steve Schneider[1] Vanessa Teague[2]

[1]Dept. of Computer Science, University of Surrey

[2]Dept. of Computer Science and Software Engineering, University of Melbourne

SnT, Luxembourg, Oct 2012

# Outline

# Outline

# Chaum's Dining Cryptographers

**Assumptions:**

- *n* cryptographers in a circle
- 0 or 1 of them paying the bill
- Honest but curious

**Goals:**

- Reveal whether anyone is paying
- Reveal nothing else

# Chaum's Dining Cryptographers

Assumptions:

- *n* cryptographers in a circle
- 0 or 1 of them paying the bill
- Honest but curious

Goals:

- Reveal whether anyone is paying
- Reveal nothing else

# Chaum's Dining Cryptographers

Assumptions:

- $n$ cryptographers in a circle
- 0 or 1 of them paying the bill
- Honest but curious

Goals:

- Reveal whether anyone is paying
- Reveal nothing else

# Chaum's Dining Cryptographers

Assumptions:

- $n$ cryptographers in a circle
- 0 or 1 of them paying the bill
- Honest but curious

Goals:

- Reveal whether anyone is paying
- Reveal nothing else

# Chaum's Dining Cryptographers

Assumptions:

- $n$ cryptographers in a circle
- 0 or 1 of them paying the bill
- Honest but curious

Goals:

- Reveal whether anyone is paying
- Reveal nothing else

# Dining Cryptographers: operation

Recipe:

1. Each adjacent pair secretly toss a coin
2. Each cryptographer says whether coin on left and coin on right gave same result
3. Must lie iff paying

# Dining Cryptographers: operation

Recipe:

1. Each adjacent pair secretly toss a coin
2. Each cryptographer says whether coin on left and coin on right gave same result
3. Must lie iff paying

# Dining Cryptographers: operation

Recipe:

1. Each adjacent pair secretly toss a coin
2. Each cryptographer says whether coin on left and coin on right gave same result
3. Must lie iff paying

# Outline

# Extending to multiple payers

Assumptions/goals:

- Now any number might be paying
- We want to know how many

Recipe:

1. Each adjacent pair secretly throw a die (up to 6 payers)
2. Each cryptographer sums left and right *modulo* 6
3. Add 1 iff paying

This now reveals how many said yes; still needs honest-but-curious model

# Extending to multiple payers

Assumptions/goals:

- Now any number might be paying
- We want to know how many

Recipe:

1. Each adjacent pair secretly throw a die (up to 6 payers)
2. Each cryptographer sums left and right *modulo* 6
3. Add 1 iff paying

This now reveals how many said yes; still needs honest-but-curious model

# Extending to multiple payers

Assumptions/goals:

- Now any number might be paying
- We want to know how many

Recipe:

1. Each adjacent pair secretly throw a die (up to 6 payers)
2. Each cryptographer sums left and right *modulo* 6
3. Add 1 iff paying

This now reveals how many said yes; still needs honest-but-curious model

# Extending to multiple payers

Assumptions/goals:

- Now any number might be paying
- We want to know how many

Recipe:

1. Each adjacent pair secretly throw a die (up to 6 payers)
2. Each cryptographer sums left and right *modulo* 6
3. Add 1 iff paying

This now reveals how many said yes; still needs honest-but-curious model

# Extending to multiple payers

Assumptions/goals:

- Now any number might be paying
- We want to know how many

Recipe:

1. Each adjacent pair secretly throw a die (up to 6 payers)
2. Each cryptographer sums left and right *modulo* 6
3. Add 1 iff paying

This now reveals how many said yes; still needs honest-but-curious model

# Extending to multiple payers

Assumptions/goals:

- Now any number might be paying
- We want to know how many

Recipe:

1. Each adjacent pair secretly throw a die (up to 6 payers)
2. Each cryptographer sums left and right *modulo* 6
3. Add 1 iff paying

This now reveals how many said yes; still needs honest-but-curious model

# Outline

- Coins and Dice: Dining Cryptographers
- Cards: Dating and Unanimity Protocols
  - Bennett's Dating Protocol
  - Extending to Three Players
- Envelopes: Secret Santa Protocols
- Envelopes: Vetoes and Threshold Voting Protocols
- Formal Analysis

# Bennett's dating protocol

Goals:

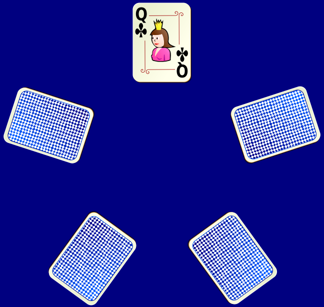
- Alice and Bob find out if both want to go on a date
- Unrequited love is terribly embarrassing

# Bennett's dating protocol

Goals:

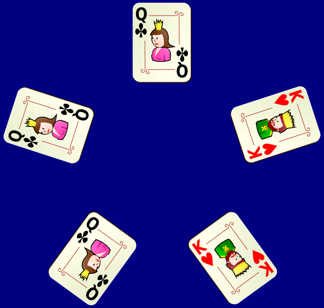
- Alice and Bob find out if both want to go on a date
- Unrequited love is terribly embarrassing

# Bennett's dating protocol: the details



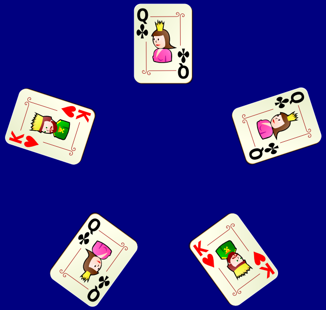
QQK = yes QKQ = no

# Bennett's dating protocol: the details



QQK = yes    QKQ = no

# Bennett's dating protocol: the details



QQK = yes QKQ = no

# Outline

# Extending to three players

Goal:

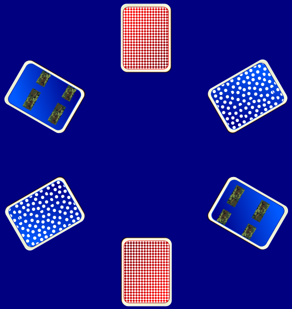
- Zero knowledge group unanimity
- Probably not suitable for dates

# Extending to three players

Goal:

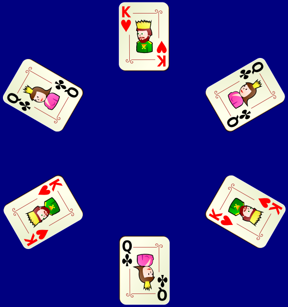
- Zero knowledge group unanimity
- Probably not suitable for dates

# Three player veto protocol



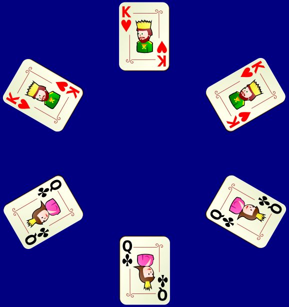
K near = yes    Q near = no

# Three player veto protocol



K near = yes    Q near = no

# Three player veto protocol



K near = yes Q near = no

# Outline

Coins and Dice: Dining Cryptographers

Cards: Dating and Unanimity Protocols

Envelopes: Secret Santa Protocols
- The Secret Santa Problem
- The Father Cryptmas Protocol
- The Faster Crassmas Protocol

Envelopes: Vetoes and Threshold Voting Protocols

Formal Analysis

# Secret Santa problem

Goals:

- Each person buys a gift for someone else
- Gift mapping should be a derangement or a cycle
- Givers are anonymous; receivers aren't

Ingredients:

- Thick envelopes
- Thick card

# Secret Santa problem

Goals:

- Each person buys a gift for someone else
- Gift mapping should be a derangement or a cycle
- Givers are anonymous; receivers aren't

Ingredients:

- Thick envelopes
- Thick card

# Secret Santa problem

Goals:

- Each person buys a gift for someone else
- Gift mapping should be a derangement or a cycle
- Givers are anonymous; receivers aren't

Ingredients:

- Thick envelopes
- Thick card

# Secret Santa problem

Goals:

- Each person buys a gift for someone else
- Gift mapping should be a derangement or a cycle
- Givers are anonymous; receivers aren't

Ingredients:

- Thick envelopes
- Thick card

# Secret Santa problem

Goals:

- Each person buys a gift for someone else
- Gift mapping should be a derangement or a cycle
- Givers are anonymous; receivers aren't

Ingredients:

- Thick envelopes
- Thick card

# Outline

- Coins and Dice: Dining Cryptographers
- Cards: Dating and Unanimity Protocols
- Envelopes: Secret Santa Protocols
  - The Secret Santa Problem
  - The Father Cryptmas Protocol
  - The Faster Crassmas Protocol
- Envelopes: Vetoes and Threshold Voting Protocols
- Formal Analysis

# The Father Cryptmas Protocol

Each person should:

1. Take an envelope and sign the front
2. Sign a card, insert facing forwards
3. Close the envelope but don't seal it

# The Father Cryptmas Protocol

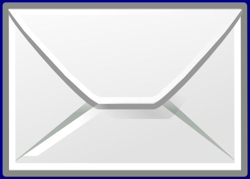
Each person should:

1. Take an envelope and sign the front
2. Sign a card, insert facing forwards
3. Close the envelope but don't seal it

# The Father Cryptmas Protocol

Each person should:

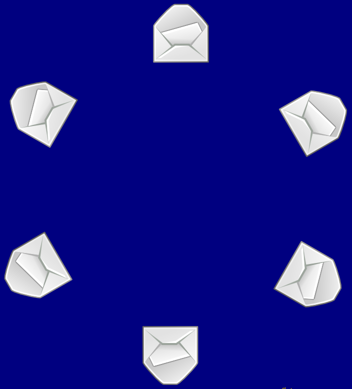
1. Take an envelope and sign the front
2. Sign a card, insert facing forwards
3. Close the envelope but don't seal it



# The Father Cryptmas Protocol: shuffling

Now shuffle and redistribute:

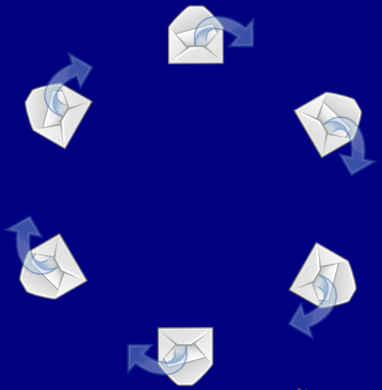
1. Shuffle face down, lay in circle, open flaps
2. Slide cards out, and move them clockwise
3. Seal, shuffle, distribute
4. Open secretly, buy gift for person named on card



# The Father Cryptmas Protocol: shuffling

Now shuffle and redistribute:

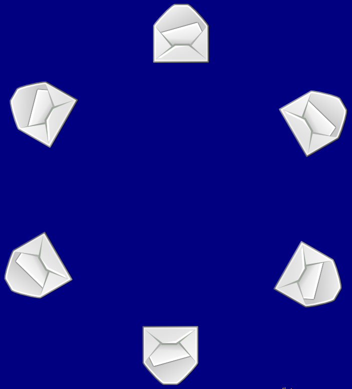
1. Shuffle face down, lay in circle, open flaps
2. Slide cards out, and move them clockwise
3. Seal, shuffle, distribute
4. Open secretly, buy gift for person named on card



# The Father Cryptmas Protocol: shuffling

Now shuffle and redistribute:

1. Shuffle face down, lay in circle, open flaps
2. Slide cards out, and move them clockwise
3. Seal, shuffle, distribute
4. Open secretly, buy gift for person named on card



# The Father Cryptmas Protocol: shuffling

Now shuffle and redistribute:

1. Shuffle face down, lay in circle, open flaps
2. Slide cards out, and move them clockwise
3. Seal, shuffle, distribute
4. Open secretly, buy gift for person named on card

# Outline

# The Faster Crassmas Protocol (Secret Scrooge)

Each person should:

1. Take an envelope and sign the front
2. Insert a €50 note
3. Close the envelope but don't seal it
4. Shuffle, move notes, reshuffle, open
5. Optionally include cards: they move the other way

Advantages:

- Exact parity of presents
- Saves buying the presents

# The Faster Crassmas Protocol (Secret Scrooge)

Each person should:

1. Take an envelope and sign the front
2. Insert a €50 note
3. Close the envelope but don't seal it
4. Shuffle, move notes, reshuffle, open
5. Optionally include cards: they move the other way

Advantages:

- Exact parity of presents
- Saves buying the presents

# The Faster Crassmas Protocol (Secret Scrooge)

Each person should:

1. Take an envelope and sign the front
2. Insert a €50 note
3. Close the envelope but don't seal it
4. Shuffle, move notes, reshuffle, open
5. Optionally include cards: they move the other way

Advantages:

- Exact parity of presents
- Saves buying the presents

# The Faster Crassmas Protocol (Secret Scrooge)

Each person should:

1. Take an envelope and sign the front
2. Insert a €50 note
3. Close the envelope but don't seal it
4. Shuffle, move notes, reshuffle, open
5. Optionally include cards: they move the other way

Advantages:

- Exact parity of presents
- Saves buying the presents

# The Faster Crassmas Protocol (Secret Scrooge)

Each person should:

1. Take an envelope and sign the front
2. Insert a €50 note
3. Close the envelope but don't seal it
4. Shuffle, move notes, reshuffle, open
5. Optionally include cards: they move the other way


Advantages:

- Exact parity of presents
- Saves buying the presents

# The Faster Crassmas Protocol (Secret Scrooge)

Each person should:

1. Take an envelope and sign the front
2. Insert a €50 note
3. Close the envelope but don't seal it
4. Shuffle, move notes, reshuffle, open
5. Optionally include cards: they move the other way


Advantages:

- Exact parity of presents
- Saves buying the presents

# The Faster Crassmas Protocol (Secret Scrooge)

Each person should:

1. Take an envelope and sign the front
2. Insert a €50 note
3. Close the envelope but don't seal it
4. Shuffle, move notes, reshuffle, open
5. Optionally include cards: they move the other way


Advantages:

- Exact parity of presents
- Saves buying the presents

# The Faster Crassmas Protocol (Secret Scrooge)

Each person should:

1. Take an envelope and sign the front
2. Insert a €50 note
3. Close the envelope but don't seal it
4. Shuffle, move notes, reshuffle, open
5. Optionally include cards: they move the other way

Advantages:

- Exact parity of presents
- Saves buying the presents

# Outline

# Veto problem

Goals:

- Each person casts a YES or a NO vote
- Any NO vote vetoes the motion
- We want to reveal only whether the motion was carried

Ingredients:

- Thick envelopes
- Thick card
- Cloth bag

# Veto problem

Goals:

- Each person casts a YES or a NO vote
- Any NO vote vetoes the motion
- We want to reveal only whether the motion was carried

Ingredients:

- Thick envelopes
- Thick card
- Cloth bag

# Veto problem

Goals:

- Each person casts a YES or a NO vote
- Any NO vote vetoes the motion
- We want to reveal only whether the motion was carried

Ingredients:

- Thick envelopes
- Thick card
- Cloth bag

# Veto problem

Goals:

- Each person casts a YES or a NO vote
- Any NO vote vetoes the motion
- We want to reveal only whether the motion was carried

Ingredients:

- Thick envelopes
- Thick card
- Cloth bag

# Veto problem

Goals:

- Each person casts a YES or a NO vote
- Any NO vote vetoes the motion
- We want to reveal only whether the motion was carried

Ingredients:

- Thick envelopes
- Thick card
- Cloth bag

# Veto problem

Goals:

- Each person casts a YES or a NO vote
- Any NO vote vetoes the motion
- We want to reveal only whether the motion was carried

Ingredients:

- Thick envelopes
- Thick card
- Cloth bag

# Threshold voting problem

Goals:

- Each person casts a YES or a NO vote
- Motion carried if YES votes exceed threshold $k$
- We want to reveal only whether the motion was carried

Ingredients:

- Thick envelopes
- Thick card
- Velcro tabs

# Threshold voting problem

Goals:

- Each person casts a YES or a NO vote
- Motion carried if YES votes exceed threshold $k$
- We want to reveal only whether the motion was carried

Ingredients:

- Thick envelopes
- Thick card
- Velcro tabs

# Threshold voting problem

Goals:

- Each person casts a YES or a NO vote
- Motion carried if YES votes exceed threshold $k$
- We want to reveal only whether the motion was carried

Ingredients:

- Thick envelopes
- Thick card
- Velcro tabs

# Threshold voting problem

Goals:

- Each person casts a YES or a NO vote
- Motion carried if YES votes exceed threshold $k$
- We want to reveal only whether the motion was carried

Ingredients:

- Thick envelopes
- Thick card
- Velcro tabs

# Threshold voting problem

Goals:

- Each person casts a YES or a NO vote
- Motion carried if YES votes exceed threshold $k$
- We want to reveal only whether the motion was carried

Ingredients:

- Thick envelopes
- Thick card
- Velcro tabs

# Threshold voting problem

Goals:

- Each person casts a YES or a NO vote
- Motion carried if YES votes exceed threshold $k$
- We want to reveal only whether the motion was carried

Ingredients:

- Thick envelopes
- Thick card
- Velcro tabs

# Outline

# Veto Protocol using envelopes

Starting point:

1. Each person seals an envelope
   - The envelope contains a blank card
2. Bag contains another sealed envelope
   - The envelope contains a YES card

Each person in turn:

1. puts his envelope into the bag
2. withdraws either
   - the same envelope (no veto)
   - the other envelope (veto)
3. discards it

# Veto Protocol using envelopes

Starting point:

1. Each person seals an envelope
   - The envelope contains a blank card
2. Bag contains another sealed envelope
   - The envelope contains a YES card


Each person in turn:

1. puts his envelope into the bag
2. withdraws either
   - the same envelope (no veto)
   - the other envelope (veto)
3. discards it

# Veto Protocol using envelopes

Starting point:

1. Each person seals an envelope
   - The envelope contains a blank card
2. Bag contains another sealed envelope
   - The envelope contains a YES card


Each person in turn:

1. puts his envelope into the bag
2. withdraws either
   - the same envelope (no veto)
   - the other envelope (veto)
3. discards it

# Veto Protocol using envelopes

Starting point:

1. Each person seals an envelope
   - The envelope contains a blank card
2. Bag contains another sealed envelope
   - The envelope contains a YES card


Each person in turn:

1. puts his envelope into the bag
2. withdraws either
   - the same envelope (no veto)
   - the other envelope (veto)
3. discards it

# Veto Protocol using envelopes

Starting point:

1. Each person seals an envelope
   - The envelope contains a blank card
2. Bag contains another sealed envelope
   - The envelope contains a YES card


Each person in turn:

1. puts his envelope into the bag
2. withdraws either
   - the same envelope (no veto)
   - the other envelope (veto)
3. discards it

# Veto Protocol using envelopes

At the end:

- the envelope in the bag is opened
- YES means no veto
- blank means vetoed

Discards must be shuffled or destroyed!

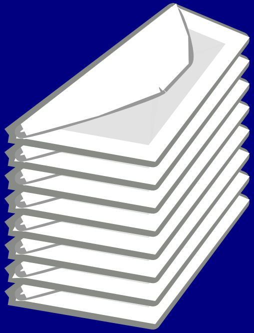
# Outline

- Coins and Dice: Dining Cryptographers
- Cards: Dating and Unanimity Protocols
- Envelopes: Secret Santa Protocols
- Envelopes: Vetoes and Threshold Voting Protocols
  - Vetoes and thresholds: the problem
  - Veto Protocol
  - Threshold Protocol
- Formal Analysis

# Threshold Protocol using envelopes

Idea:

- Rather than one envelope in the bag:
  - we need a FIFO queue of envelopes
  - to say no: add a NO, pop a YES
  - to say yes: do nothing
  - carried if $\geq k$ YESes popped



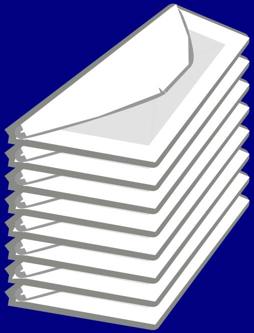
# Threshold Protocol using envelopes

**Starting point:**

- Stack of $k$ YES envelopes
- Everyone holds a NO envelope

**Each participant:**

1. takes a NO envelope
2. adds it to the top of the stack
3. takes the stack under the table
4. discards either
   - the top one (vote YES)
   - the bottom one (vote NO)
5. replaces the stack



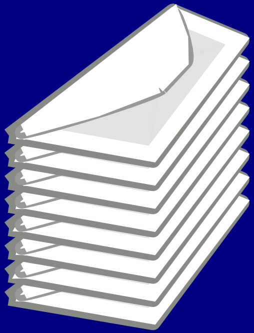
# Threshold Protocol using envelopes

Starting point:

- Stack of $k$ YES envelopes
- Everyone holds a NO envelope

Each participant:

1. takes a NO envelope
2. adds it to the top of the stack
3. takes the stack under the table
4. discards either
   - the top one (vote YES)
   - the bottom one (vote NO)
5. replaces the stack



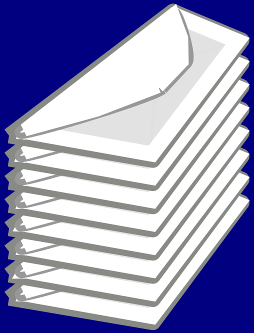
# Threshold Protocol using envelopes

Starting point:

- Stack of $k$ YES envelopes
- Everyone holds a NO envelope

Each participant:

1. takes a NO envelope
2. adds it to the top of the stack
3. takes the stack under the table
4. discards either
   - the top one (vote YES)
   - the bottom one (vote NO)
5. replaces the stack



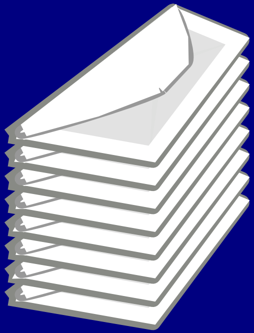
# Threshold Protocol using envelopes

Starting point:

- Stack of $k$ YES envelopes
- Everyone holds a NO envelope

Each participant:

1. takes a NO envelope
2. adds it to the top of the stack
3. takes the stack under the table
4. discards either
   - the top one (vote YES)
   - the bottom one (vote NO)
5. replaces the stack



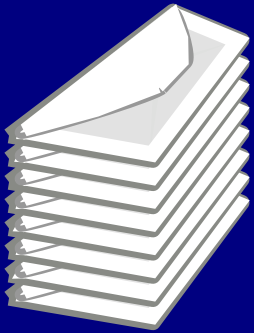
# Threshold Protocol using envelopes

Starting point:

- Stack of $k$ YES envelopes
- Everyone holds a NO envelope

Each participant:

1. takes a NO envelope
2. adds it to the top of the stack
3. takes the stack under the table
4. discards either
    - the top one (vote YES)
    - the bottom one (vote NO)
5. replaces the stack



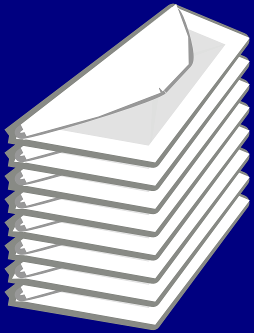
# Threshold Protocol using envelopes

Starting point:

- Stack of $k$ YES envelopes
- Everyone holds a NO envelope

Each participant:

1. takes a NO envelope
2. adds it to the top of the stack
3. takes the stack under the table
4. discards either
   - the top one (vote YES)
   - the bottom one (vote NO)
5. replaces the stack



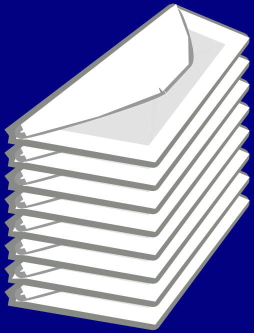
# Threshold Protocol using envelopes

Starting point:

- Stack of *k* YES envelopes
- Everyone holds a NO envelope

Each participant:

1. takes a NO envelope
2. adds it to the top of the stack
3. takes the stack under the table
4. discards either
   - the top one (vote YES)
   - the bottom one (vote NO)
5. replaces the stack



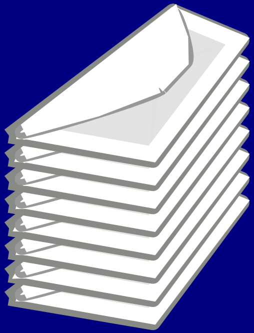
# Threshold Protocol using envelopes

Starting point:

- Stack of $k$ YES envelopes
- Everyone holds a NO envelope

Each participant:

1. takes a NO envelope
2. adds it to the top of the stack
3. takes the stack under the table
4. discards either
   - the top one (vote YES)
   - the bottom one (vote NO)
5. replaces the stack



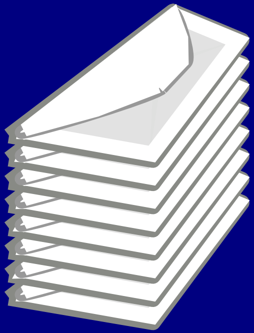
# Threshold Protocol using envelopes

Starting point:

- Stack of $k$ YES envelopes
- Everyone holds a NO envelope

Each participant:

1. takes a NO envelope
2. adds it to the top of the stack
3. takes the stack under the table
4. discards either
   - the top one (vote YES)
   - the bottom one (vote NO)
5. replaces the stack



# Threshold Protocol using envelopes

At the end:

- Open the bottom envelope
- It contains the group decision

Discards must be shuffled...

# Threshold Protocol using envelopes

At the end:

- Open the bottom envelope
- It contains the group decision

Discards must be shuffled...

# Strengthening the Threshold Protocol

Works in honest-but-curious model:

- but not a stronger attacker
- can manipulate the stack

Solution:

- add Velcro tabs to each envelope
- publicly stick envelope to top
- only allowed one ‘rip’ under the table

# Outline

# Modelling the protocols in CSP

We will model the dating protocol and the unanimity protocol in CSP:

- Build some general functions and processes:
  - Process to 'pick the cards up'
  - Function to generate rotations of a sequence
  - Process to announce a rotation non-deterministically
  - Process to allow us to control players' choices (for specification)
- Use them to model each protocol

# Modelling the protocols in CSP

We will model the dating protocol and the unanimity protocol in CSP:

- Build some general functions and processes:
  - Process to ‘pick the cards up’
  - Function to generate rotations of a sequence
  - Process to announce a rotation non-deterministically
  - Process to allow us to control players’ choices (for specification)
- Use them to model each protocol

# Modelling the protocols in CSP

We will model the dating protocol and the unanimity protocol in CSP:

- Build some general functions and processes:
  - Process to 'pick the cards up'
  - Function to generate rotations of a sequence
  - Process to announce a rotation non-deterministically
  - Process to allow us to control players' choices (for specification)
- Use them to model each protocol

# Collecting cards and announcing a result

$$ANNOUNCE(xs) = ANNOUNCE\_FROM(allrots(xs))$$

$$ANNOUNCE\_FROM(xss) = \bigsqcap_{xs \in xss} announce!xs \rightarrow Stop$$

$$COLLECTING(xs, 0, cur) = rotate \rightarrow ANNOUNCE(xs)$$

$$COLLECTING(xs, rem, cur) =$$

$$place.cur?X \rightarrow COLLECTING(xs \frown \langle X \rangle, rem - 1, cur + 1)$$

# Outline

Coins and Dice: Dining Cryptographers

Cards: Dating and Unanimity Protocols

Envelopes: Secret Santa Protocols

Envelopes: Vetoes and Threshold Voting Protocols

Formal Analysis
- Common Issues
- Dating Protocol
- Unanimity Protocol

# Modelling the dating protocol

$$
\begin{aligned}
P1\_DATE &= accept.1 \rightarrow place.1.Q \rightarrow place.2.K \rightarrow Stop \\
&\quad \square\; veto.1 \rightarrow place.1.K \rightarrow place.2.Q \rightarrow Stop \\
P2\_DATE &= accept.2 \rightarrow place.3.K \rightarrow place.4.Q \rightarrow Stop \\
&\quad \square\; veto.2 \rightarrow place.3.Q \rightarrow place.4.K \rightarrow Stop
\end{aligned}
$$

Note:

- Players accept or veto, then place cards accordingly
- No cheating allowed: honest-but-curious model

# Modelling the dating protocol

$$
\begin{aligned}
P1\_DATE &= accept.1 \rightarrow place.1.Q \rightarrow place.2.K \rightarrow Stop \\
&\;\square\; veto.1 \rightarrow place.1.K \rightarrow place.2.Q \rightarrow Stop \\
P2\_DATE &= accept.2 \rightarrow place.3.K \rightarrow place.4.Q \rightarrow Stop \\
&\;\square\; veto.2 \rightarrow place.3.Q \rightarrow place.4.K \rightarrow Stop
\end{aligned}
$$

Note:

- Players accept or veto, then place cards accordingly
- No cheating allowed: honest-but-curious model

# Modelling the dating protocol

$$
\begin{aligned}
P1\_DATE &= accept.1 \rightarrow place.1.Q \rightarrow place.2.K \rightarrow Stop \\
&\;\;\square\; veto.1 \rightarrow place.1.K \rightarrow place.2.Q \rightarrow Stop \\
P2\_DATE &= accept.2 \rightarrow place.3.K \rightarrow place.4.Q \rightarrow Stop \\
&\;\;\square\; veto.2 \rightarrow place.3.Q \rightarrow place.4.K \rightarrow Stop
\end{aligned}
$$

Note:

- Players accept or veto, then place cards accordingly
- No cheating allowed: honest-but-curious model

# Building the system

$$COLLECT\_CARDS\_DATE = COLLECTING(\langle Q \rangle, 4, 1)$$

$$DATE\_SYSTEM = (P1\_DATE \mathbin{|||} P2\_DATE) \underset{\{|place.x | x \in \{1..4\}|\}}{\|} COLLECT\_CARDS\_DATE$$

# Specifying the property

$$P2\_EVENTS = \{|accept.2, veto.2, place.3, place.4|\}$$

$$P1\_DATE\_VIEW(choices) = (DATE\_SYSTEM \underset{\{|accept, veto|\}}{\|} CONTROLS(choices)) \setminus P2\_EVENTS$$

$$P1\_DATE\_VIEW(\langle 0, 0 \rangle) =_T P1\_DATE\_VIEW(\langle 0, 1 \rangle)$$

- Player 2 handled similarly
- FDR confirms specifications hold

# Specifying the property

$$
\begin{aligned}
P2\_EVENTS &= \{|accept.2, veto.2, place.3, place.4|\} \\
P1\_DATE\_VIEW(choices) &= \\
(DATE\_SYSTEM & \underset{\{|accept, veto|\}}{\parallel} CONTROLS(choices)) \\
& \setminus P2\_EVENTS
\end{aligned}
$$

$$P1\_DATE\_VIEW(\langle 0, 0\rangle) =_T P1\_DATE\_VIEW(\langle 0, 1\rangle)$$

- Player 2 handled similarly
- FDR confirms specifications hold

# Specifying the property

$$\begin{aligned} P2\_EVENTS &= \{|accept.2, veto.2, place.3, place.4|\} \\ P1\_DATE\_VIEW(choices) &= \\ (DATE\_SYSTEM & \underset{\{|accept, veto|\}}{\|} CONTROLS(choices)) \\ & \setminus P2\_EVENTS \end{aligned}$$

$$P1\_DATE\_VIEW(\langle 0, 0\rangle) =_T P1\_DATE\_VIEW(\langle 0, 1\rangle)$$

- Player 2 handled similarly
- FDR confirms specifications hold

# Specifying the property

$$
\begin{aligned}
P2\_EVENTS &= \{|accept.2, veto.2, place.3, place.4|\} \\
P1\_DATE\_VIEW(choices) &= \\
(DATE\_SYSTEM & \underset{\{|accept, veto|\}}{\parallel} CONTROLS(choices)) \\
& \setminus P2\_EVENTS
\end{aligned}
$$

$$
P1\_DATE\_VIEW(\langle 0, 0 \rangle) =_T P1\_DATE\_VIEW(\langle 0, 1 \rangle)
$$

- Player 2 handled similarly
- FDR confirms specifications hold

# Outline

Coins and Dice: Dining Cryptographers

Cards: Dating and Unanimity Protocols

Envelopes: Secret Santa Protocols

Envelopes: Vetoes and Threshold Voting Protocols

Formal Analysis
- Common Issues
- Dating Protocol
- Unanimity Protocol

# Modelling the unanimity protocol

Almost exactly the same:

$$\begin{array}{l} \mathit{EXT\_UNANIM\_VIEW}(\mathit{choices}) = \\ \quad (\mathit{UNANIM\_SYSTEM} \underset{\{|\mathit{accept},\mathit{veto}|\}}{\parallel} \mathit{CONTROLS}(\mathit{choices})) \\ \qquad \setminus \{|\mathit{accept}, \mathit{veto}, \mathit{place}|\} \end{array}$$

- From the perspective of an external observer
- Only the final rotated arrangement visible

# Modelling the unanimity protocol

Almost exactly the same:

$$
\begin{aligned}
&\mathit{EXT\_UNANIM\_VIEW}(\mathit{choices}) = \\
&\quad (\mathit{UNANIM\_SYSTEM} \underset{\{|\mathit{accept},\mathit{veto}|\}}{\|} \mathit{CONTROLS}(\mathit{choices})) \\
&\qquad\qquad \setminus \{|\mathit{accept}, \mathit{veto}, \mathit{place}|\}
\end{aligned}
$$

- From the perspective of an external observer
- Only the final rotated arrangement visible

# Modelling the unanimity protocol

Almost exactly the same:

$$\mathit{EXT\_UNANIM\_VIEW}(\mathit{choices}) = (\mathit{UNANIM\_SYSTEM} \underset{\{|\mathit{accept},\mathit{veto}|\}}{\parallel} \mathit{CONTROLS}(\mathit{choices})) \setminus \{|\mathit{accept}, \mathit{veto}, \mathit{place}|\}$$

- From the perspective of an external observer
- Only the final rotated arrangement visible

# Specifications for the unanimity protocol

$$EXT\_UNANIM\_VIEW(\langle 0,0,0\rangle) =_T EXT\_UNANIM\_VIEW(\langle 1,1,1\rangle)$$

and, whenever $\{a,b,c\} = \{d,e,f\} = \{0,1\}$

$$EXT\_UNANIM\_VIEW(\langle a,b,c\rangle) =_T EXT\_UNANIM\_VIEW(\langle d,e,f\rangle)$$

# Specifications for the unanimity protocol

$$EXT\_UNANIM\_VIEW(\langle 0,0,0\rangle) =_T EXT\_UNANIM\_VIEW(\langle 1,1,1\rangle)$$

and, whenever $\{a,b,c\} = \{d,e,f\} = \{0,1\}$

$$EXT\_UNANIM\_VIEW(\langle a,b,c\rangle) =_T EXT\_UNANIM\_VIEW(\langle d,e,f\rangle)$$

# Arguing in a circle

```
-- These six checks succeed
assert EXT_UNANIM_VIEW(<0,0,1>) [T= EXT_UNANIM_VIEW(<0,1,0>)
assert EXT_UNANIM_VIEW(<0,1,0>) [T= EXT_UNANIM_VIEW(<0,1,1>)
assert EXT_UNANIM_VIEW(<0,1,1>) [T= EXT_UNANIM_VIEW(<1,0,0>)
assert EXT_UNANIM_VIEW(<1,0,0>) [T= EXT_UNANIM_VIEW(<1,0,1>)
assert EXT_UNANIM_VIEW(<1,0,1>) [T= EXT_UNANIM_VIEW(<1,1,0>)
assert EXT_UNANIM_VIEW(<1,1,0>) [T= EXT_UNANIM_VIEW(<0,0,1>)

-- These two checks fail
assert EXT_UNANIM_VIEW(<0,0,0>) [T= EXT_UNANIM_VIEW(<0,0,1>)
assert EXT_UNANIM_VIEW(<0,0,1>) [T= EXT_UNANIM_VIEW(<0,0,0>)
```

# Conclusions

Conclusions:

- Several new protocols
  - Give reasonable security in social contexts
  - Don’t require any crypto or electronics
- Formal analysis of two protocols

# Conclusions

Conclusions:

- Several new protocols
  - Give reasonable security in social contexts
  - Don’t require any crypto or electronics
- Formal analysis of two protocols