

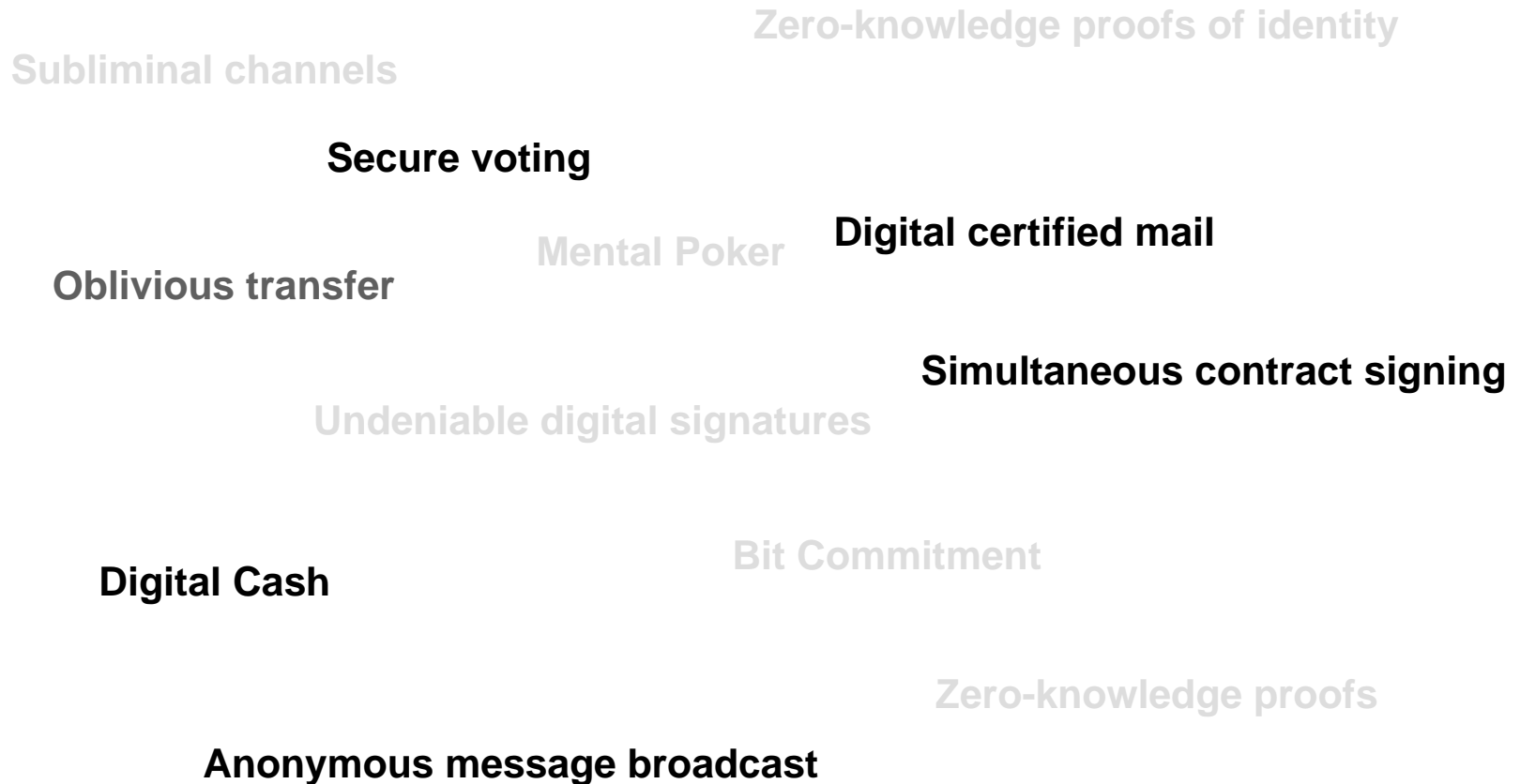
NIS/CpE 691A

Information System Security

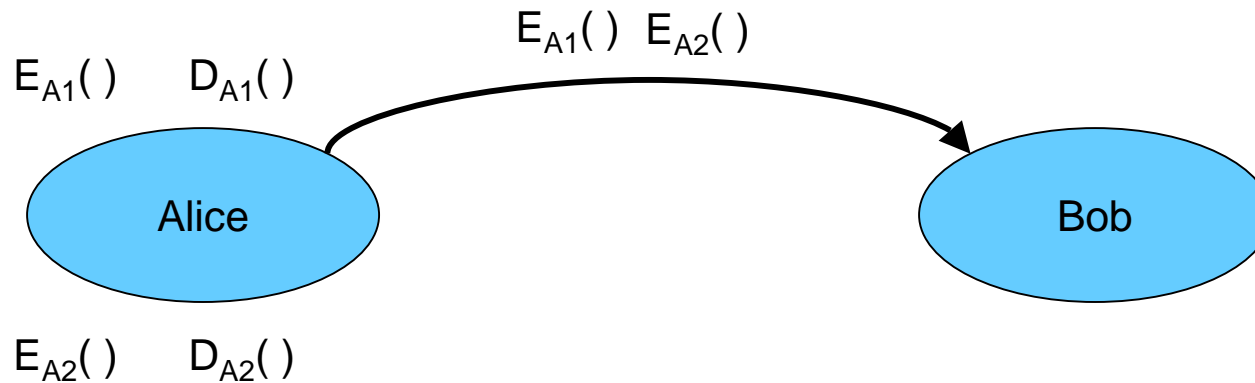
**Stevens Institute of Technology
Spring 2006**

Class 11 – 4/6/06

More Security Protocols

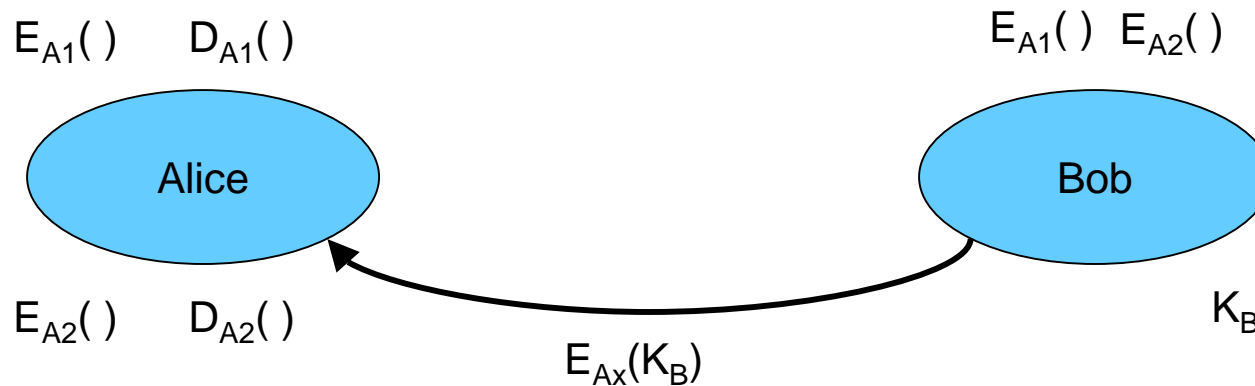


Oblivious Transfer (review)



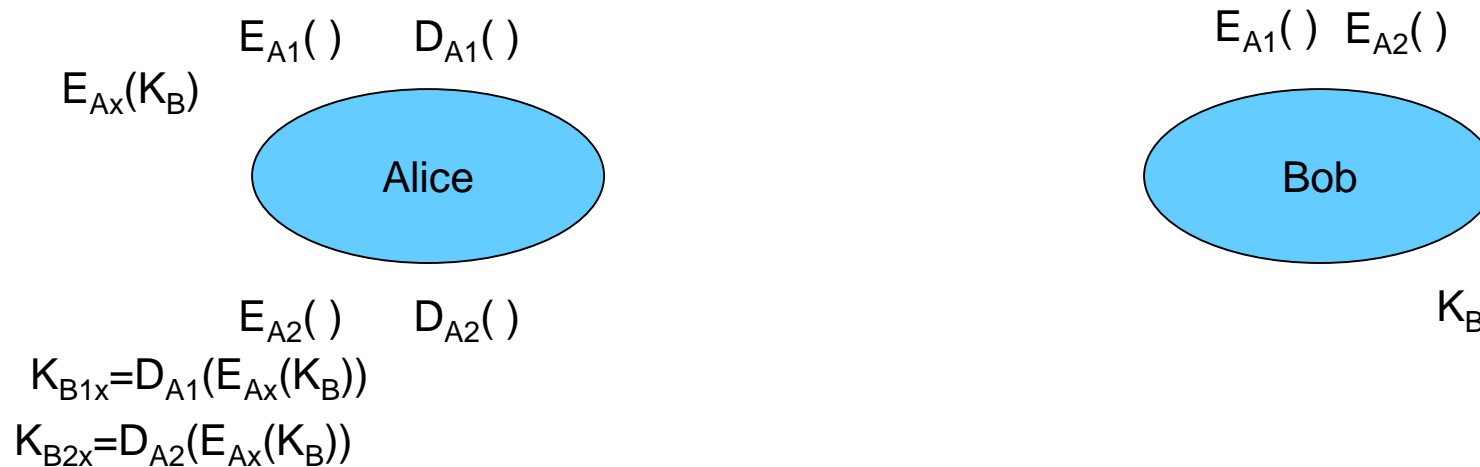
- Alice generates two PKC key pairs and sends the public half of each to Bob

Oblivious Transfer (review)



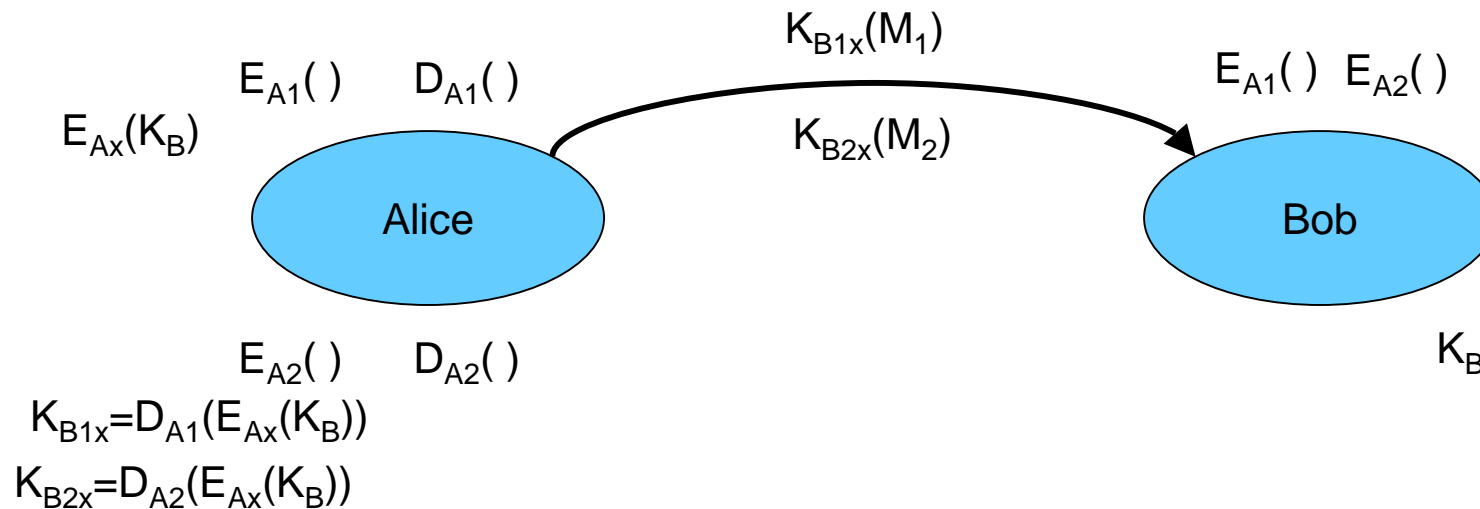
- Alice generates two PKC key pairs and sends the public half of each to Bob
- Bob generates a key for a symmetric cryptosystem, secretly picks one of Alices public keys and sends the encrypted key to Alice

Oblivious Transfer (review)



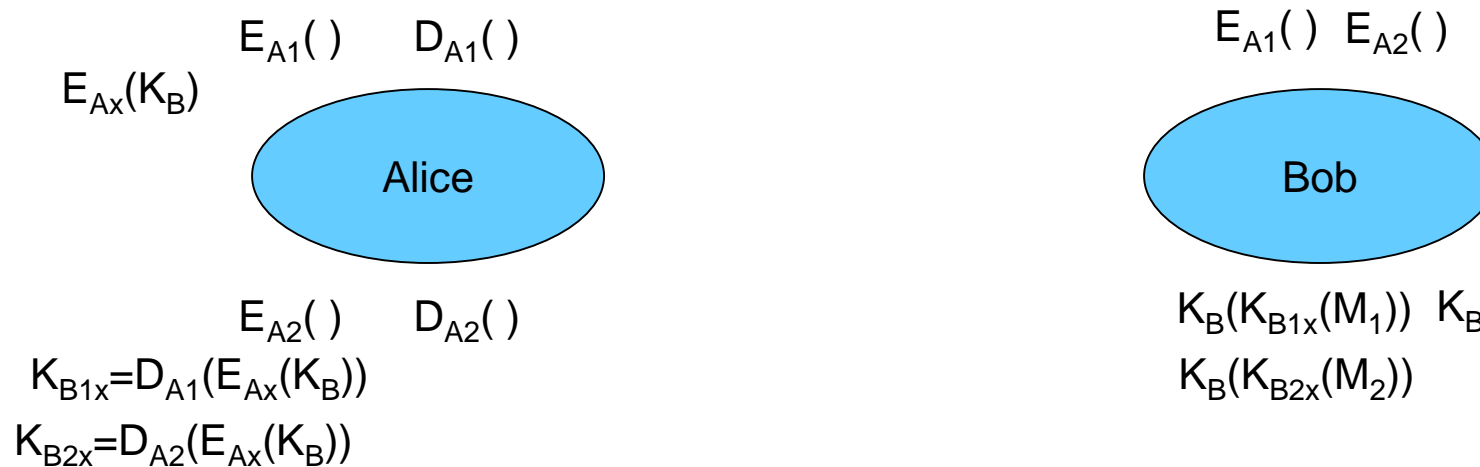
- Alice generates two PKC key pairs and sends the public half of each to Bob
- Bob generates a key for a symmetric cryptosystem, secretly picks one of Alices public keys and sends the encrypted key to Alice
- Not knowing which of her private keys to use, Alice decrypts Bob's message with both - providing a valid key and gibberish, but not knowing which is which

Oblivious Transfer (review)



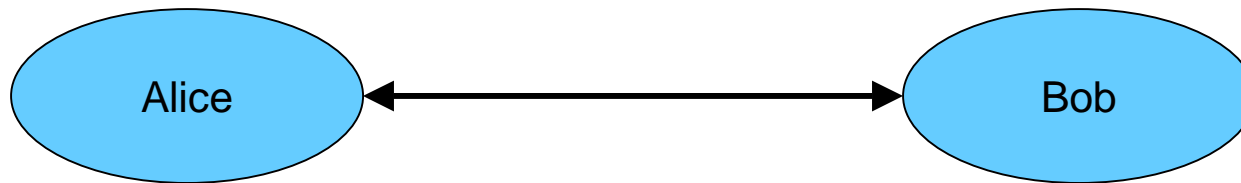
- Alice generates two PKC key pairs and sends the public half of each to Bob
- Bob generates a key for a symmetric cryptosystem, secretly picks one of Alices public keys and sends the encrypted key to Alice
- Not knowing which of her private keys to use, Alice decrypts Bob's message with both - providing a valid key and gibberish, but not knowing which is which
- Alice sends two messages to Bob, each encrypted with one of the keys generated

Oblivious Transfer (review)



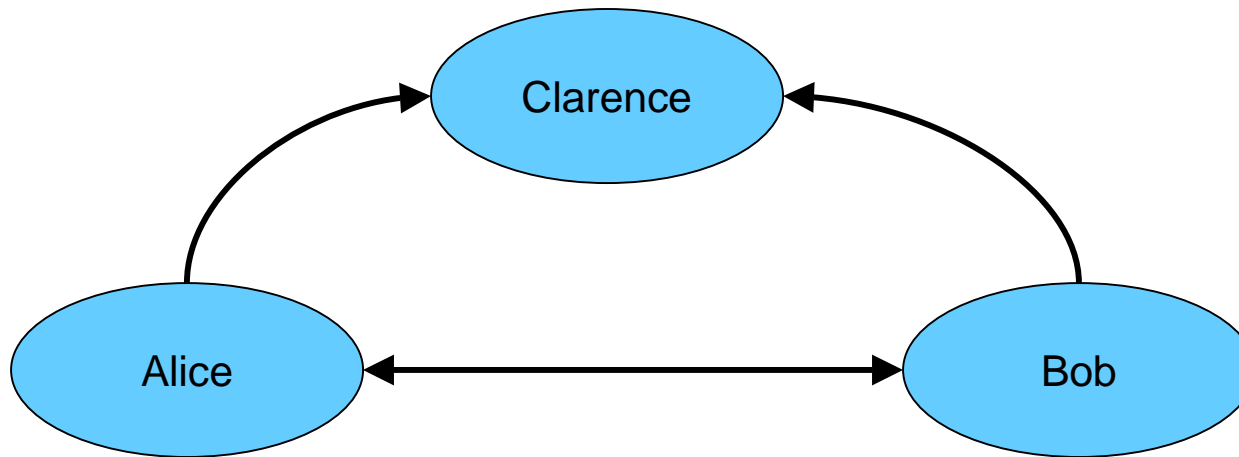
- Alice generates two PKC key pairs and sends the public half of each to Bob
- Bob generates a key for a symmetric cryptosystem, secretly picks one of Alices public keys and sends the encrypted key to Alice
- Not knowing which of her private keys to use, Alice decrypts Bob's message with both - providing a valid key and gibberish, but not knowing which is which
- Alice sends two messages to Bob, each encrypted with one of the keys generated
- Bob decrypts both messages with his key. Only one successfully. Alice is oblivious to which was received

Simultaneous Contract Signing



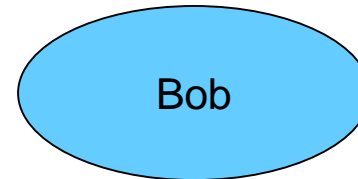
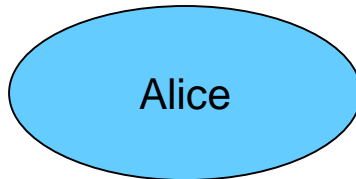
- Alice and Bob want to sign a contract, but neither wants to sign unless the other has signed.

Simultaneous Contract Signing



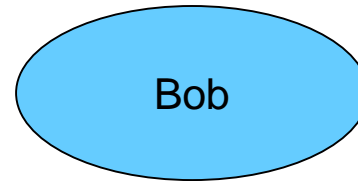
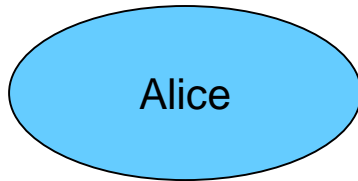
- Alice and Bob want to sign a contract, but neither wants to sign unless the other has signed.
- With a trusted arbitrator, Clarence, this is a simple process:
 - Alice and Bob sign a copy of contract and send it to the arbitrator
 - The arbitrator tells both that the other has signed
 - Alice signs two copies of contract and sends them to Bob
 - Bob signs both copies, keeps one, and returns the other to Alice
 - The arbitrator destroys the copies with one signature
- Alice and Bob are protected, since the arbitrator can provide either with a copy that only needs the requestor's signature

Simultaneous Contract Signing Without an Arbitrator



- Alice and Bob want to sign a contract, but neither wants to sign unless the other has signed. Clarence's fees are too high, so they decide to resort to a security protocol

Simultaneous Contract Signing Without an Arbitrator



$K_{AL1}, K_{AR1},$
...
 K_{AL100}, K_{AR100}

$K_{BL1}, K_{BR1},$
...
 K_{BL100}, K_{BR100}

- Alice and Bob both generate 200 keys for a symmetric cryptosystem (DES). The keys are grouped into 100 pairs

Simultaneous Contract Signing Without an Arbitrator



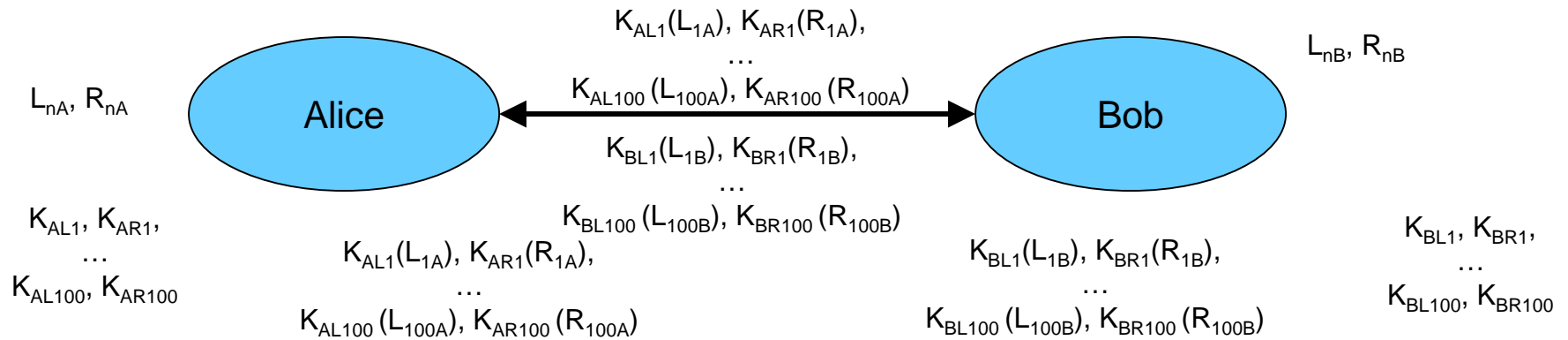
- Alice and Bob also generate 100 pairs of messages, L_n and R_n
 - The messages are:
 - “*right half of n^{th} signature, signature_of_contract, timestamp*”, and
 - “*left half of n^{th} signature, signature_of_contract, timestamp*”
 - The contract is considered signed by Bob(Alice) if Alice(Bob) can produce a message that consists of Bob’s(Alice’s) L_n and R_n message for some n

Simultaneous Contract Signing Without an Arbitrator



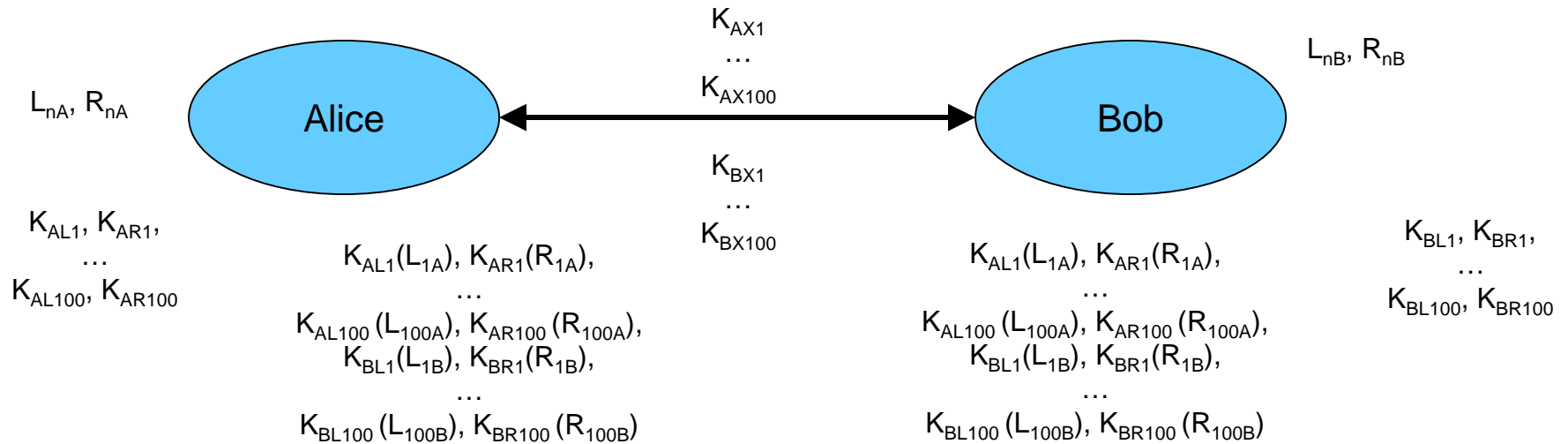
- Alice and Bob encrypt their message pairs with each key pair, respectively

Simultaneous Contract Signing Without an Arbitrator



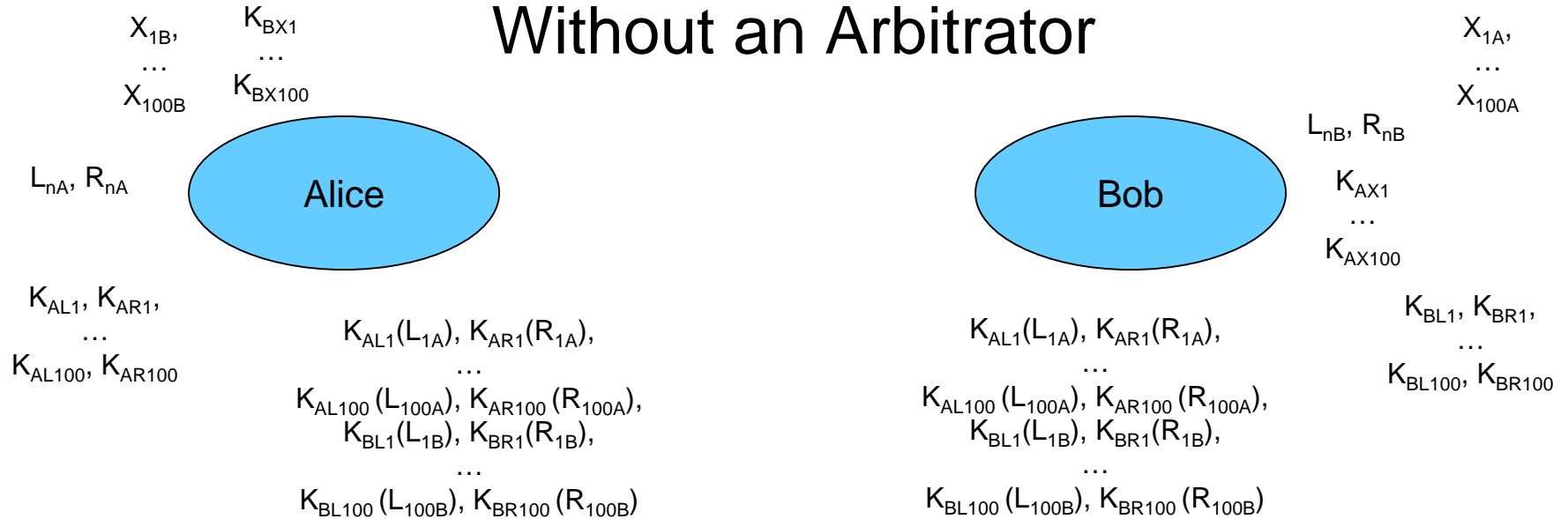
- Alice and Bob send each other the encrypted messages

Simultaneous Contract Signing Without an Arbitrator



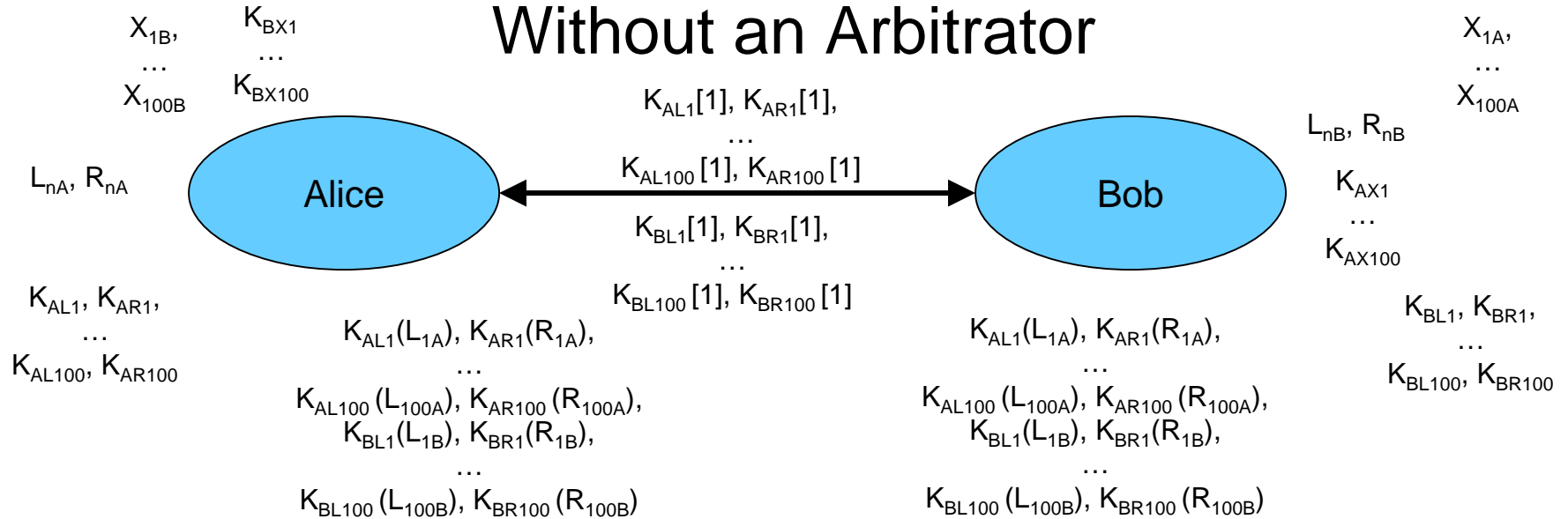
- Alice and Bob send each other each key pair, using oblivious transfer for each pair.
- Alice and Bob now each have one key in each pair, but the sender does not know which

Simultaneous Contract Signing Without an Arbitrator



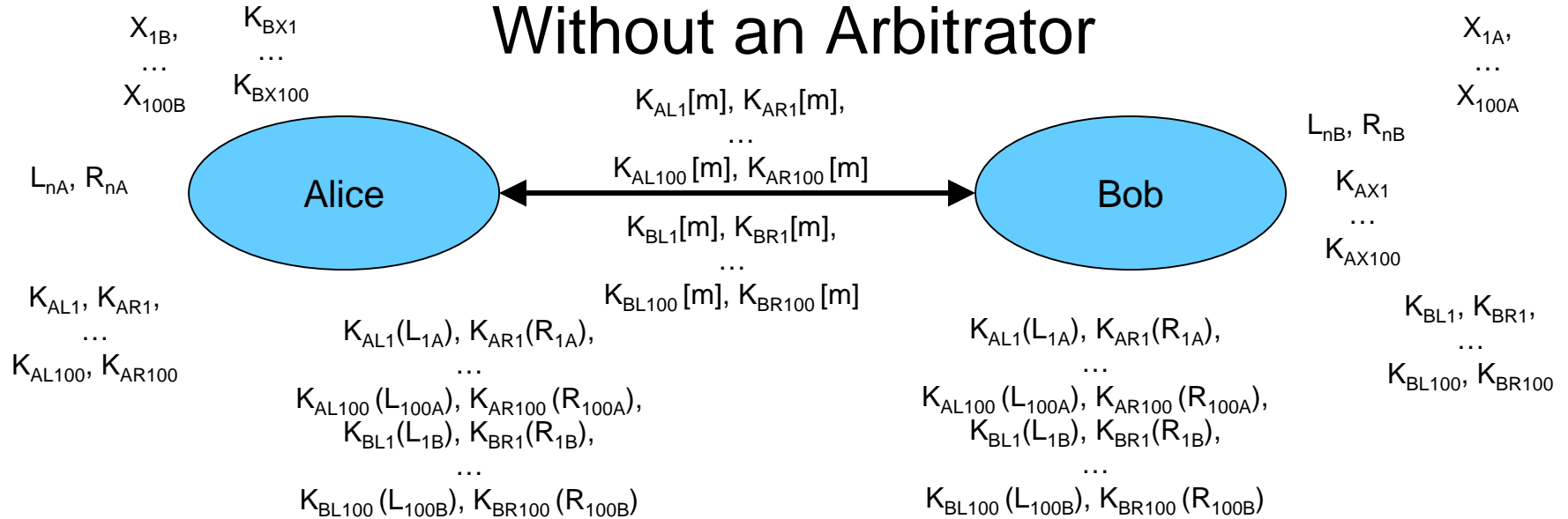
- Alice and Bob decrypt the message halves they have using the appropriate key half.
- They make sure the decrypted messages they have are valid

Simultaneous Contract Signing Without an Arbitrator



- Alice and Bob each send each other the first bits of all 200 keys
- Since each has an entire key for half of each message, this bit can be verified for half the message halves.

Simultaneous Contract Signing Without an Arbitrator



- Alice and Bob repeat the previous step for each of the m bits of the keys
- Since each has an entire key for half of each message, this bit can be verified for half the message halves.
- When all m bits have been received for all message halves, both Alice and Bob have the required L_n and R_n message (for all n , in this case)

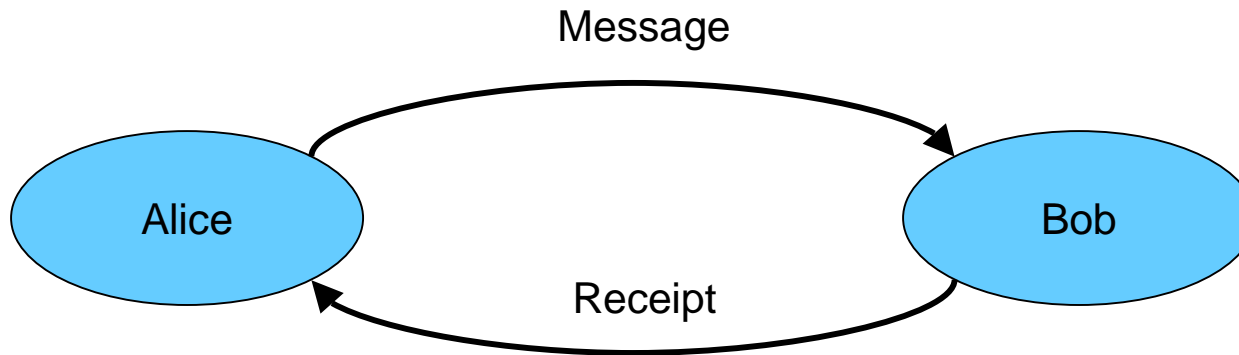
Simultaneous Contract Signing Without an Arbitrator

- Without loss of generality, assume that Alice wants to cheat.
- She could send random strings instead of the encrypted messages and the oblivious transfer strings. Bob would notice this when he tried to decrypt the message halves and wouldn't finish the protocol, denying Alice a signed contract.
- Alice could transmit right half messages correctly but send junk for the left half. However, since there are 100 message sets, this would have a probability of 2^{-100} of succeeding
- Alice could try to send random bits for the m bits of the key. But Bob already has half of the bits and Alice doesn't know which half due to the oblivious transfer, so this won't work.
- Alice could terminate the protocol early when the m bits are being exchanged. However, Bob has as many key bits as Alice does, so this doesn't help her.
UNLESS:

Simultaneous Contract Signing Without an Arbitrator

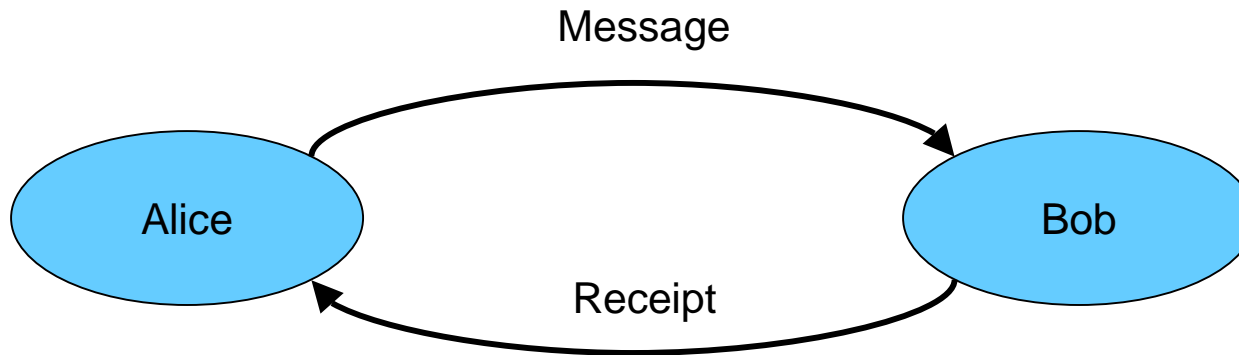
- Without loss of generality, assume that Alice wants to cheat.
- She could send random strings instead of the encrypted messages and the oblivious transfer strings. Bob would notice this when he tried to decrypt the message halves and wouldn't finish the protocol, denying Alice a signed contract.
- Alice could transmit right half messages correctly but send junk for the left half. However, since there are 100 message sets, this would have a probability of 2^{-100} of succeeding
- Alice could try to send random bits for the m bits of the key. But Bob already has half of the bits and Alice doesn't know which half due to the oblivious transfer, so this won't work.
- Alice could terminate the protocol early when the m bits are being exchanged. However, Bob has as many key bits as Alice does, so this doesn't help her.
UNLESS:
 - What if Alice has more computing power than Bob?
 - She could stop when there were 30 bits left to the keys. Both have to solve the same problem - guessing 30 bits when $m-30$ are known.
 - With more computing power, Alice might have a time advantage over Bob - she can show that the contract was signed and he defaulted on a delivery, while he cannot force her to meet her terms of the contract.

Digital Certified Mail



- Alice is serving Bob with a legal notice by Digital Certified Mail.
- Alice wants Bob to acknowledge receiving the message before he reads it (if Bob knew what was in the message, he might say he never got it)

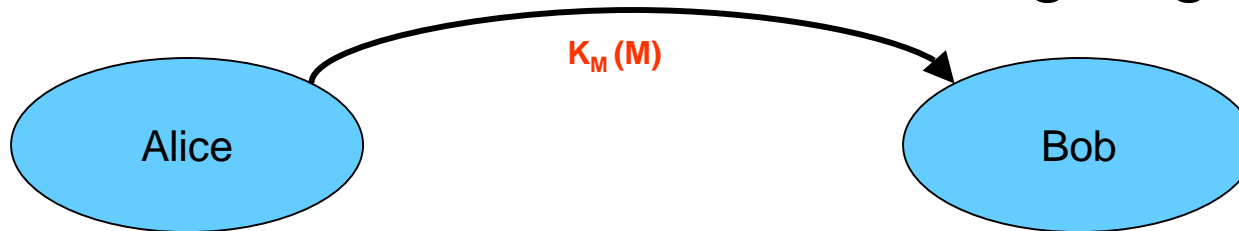
Digital Certified Mail



- On the surface, this looks like a Simultaneous Contract Signing protocol could be used:
 - Alice sends Bob key halves, Bob sends receipt halves
 - But: Cheating is prevented by the oblivious transfer. In this case, Bob wouldn't know if he has received half a key - it is indistinguishable from a random string.

Digital Certified Mail - Differences from Simultaneous Contract Signing

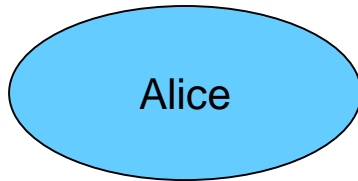
M, K_M



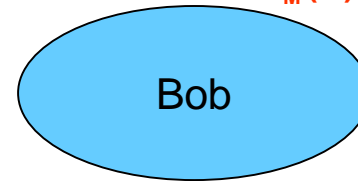
- Alice encrypts her certified mail with a random key and sends the encrypted message to Bob

Digital Certified Mail - Differences from Simultaneous Contract Signing

M, K_M



$K_M(M)$



$K_{A1}, K_{A1+K_M},$

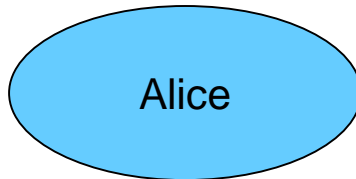
...

K_{A100}, K_{A100+K_M}

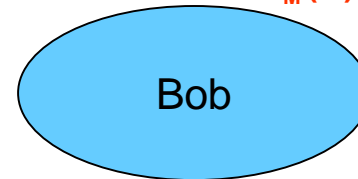
- Alice generates 100 pairs of keys for a symmetric cryptosystem (DES). The first key of each pair is random, the second is the first key XOR the message key pairs

Digital Certified Mail - Differences from Simultaneous Contract Signing

M, K_M



$K_M(M)$



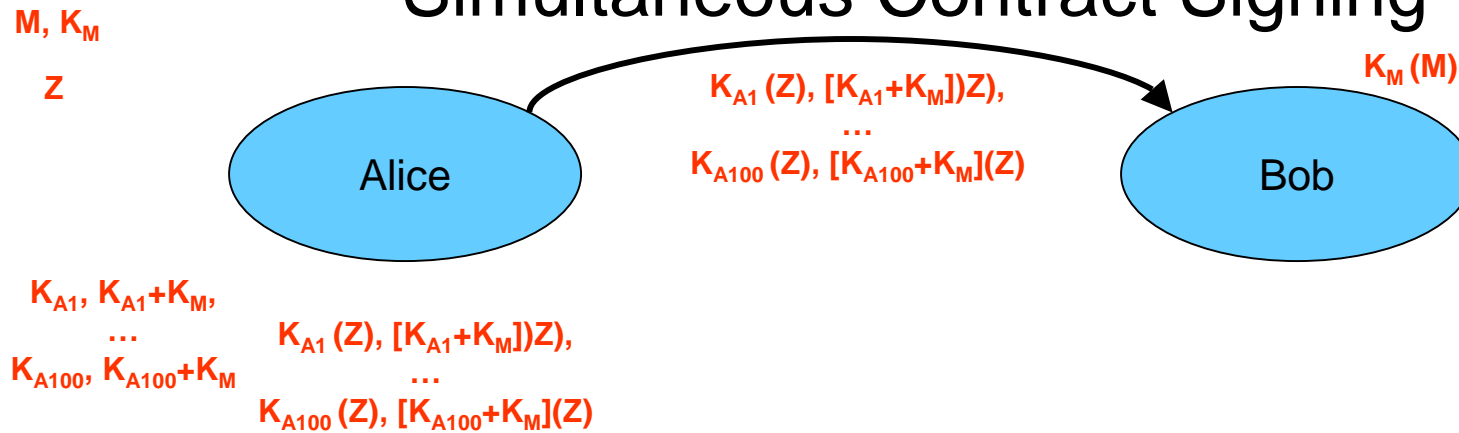
$K_{A1}, K_{A1+K_M},$

...

K_{A100}, K_{A100+K_M}

- Alice generates 100 pairs of keys for a symmetric cryptosystem (DES). The first key of each pair is random, the second is the first key XOR the message key pairs

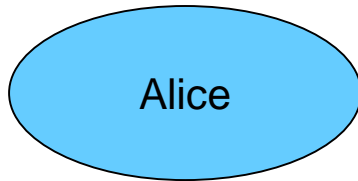
Digital Certified Mail - Differences from Simultaneous Contract Signing



- Alice generates a dummy message, Z , and encrypts it with each of her 200 keys
- She sends the encrypted messages to Bob

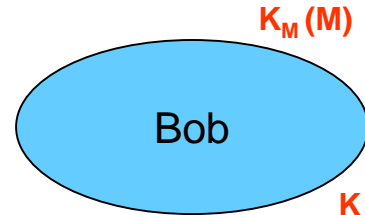
Digital Certified Mail - Differences from Simultaneous Contract Signing

M, K_M
 Z



$K_{A1}, K_{A1+K_M},$
...
 K_{A100}, K_{A100+K_M}

$K_{A1}(Z), [K_{A1}+K_M](Z),$
...
 $K_{A100}(Z), [K_{A100}+K_M](Z)$



$K_M(M)$

$K_{BL1}, K_{BR1},$
...
 $K_{BL100}, K_{BR100},$

$K_{A1}(Z), [K_{A1}+K_M](Z),$
...
 $K_{A100}(Z), [K_{A100}+K_M](Z)$

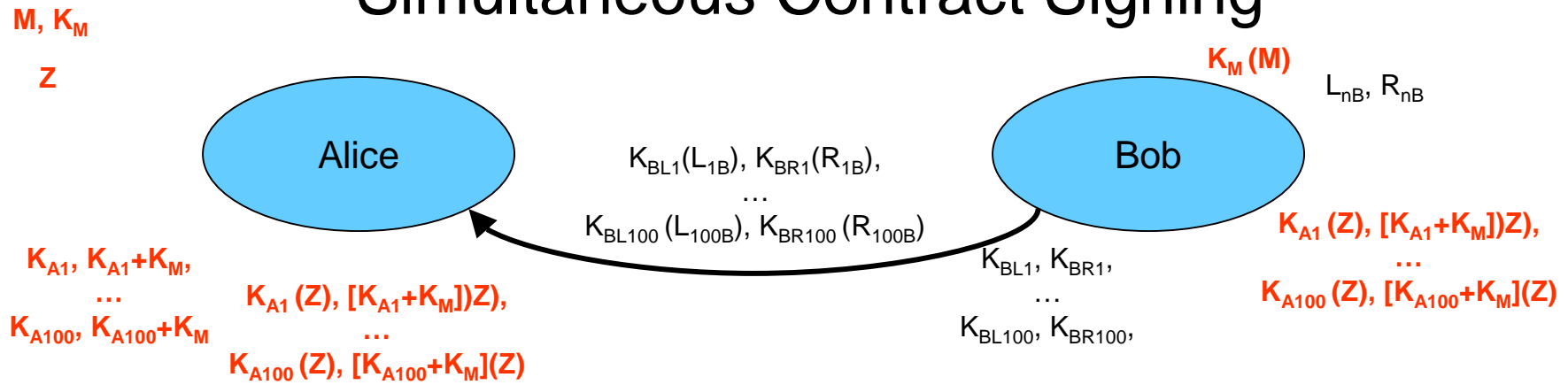
- Bob generates 100 pairs of random keys

Digital Certified Mail - Differences from Simultaneous Contract Signing



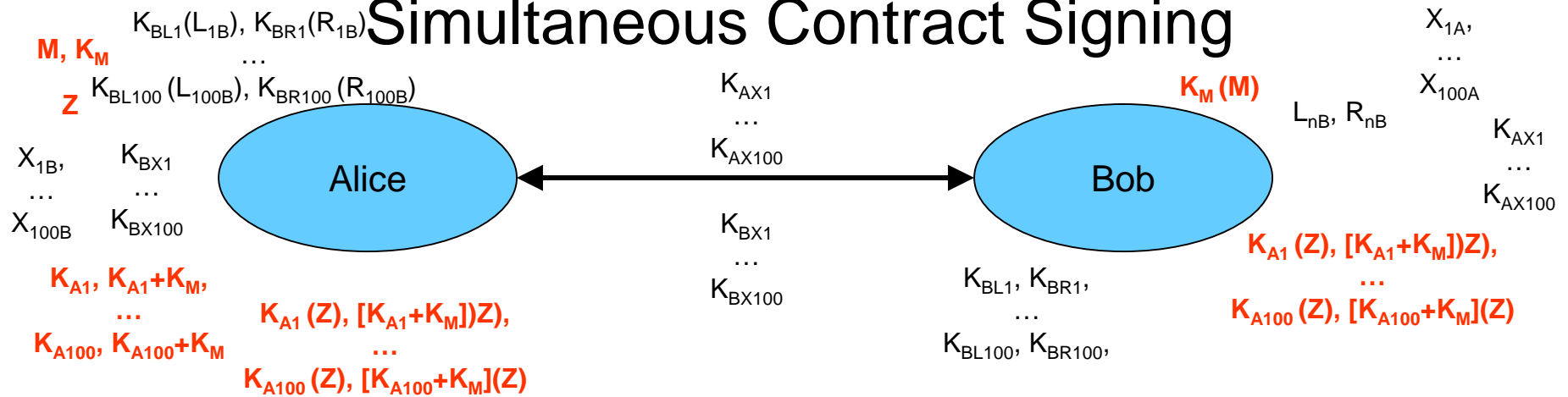
- Bob generates 100 receipt pairs:
 - “This is the left half of my i^{th} receipt + validation”
 - “This is the right half of my i^{th} receipt + validation”

Digital Certified Mail - Differences from Simultaneous Contract Signing



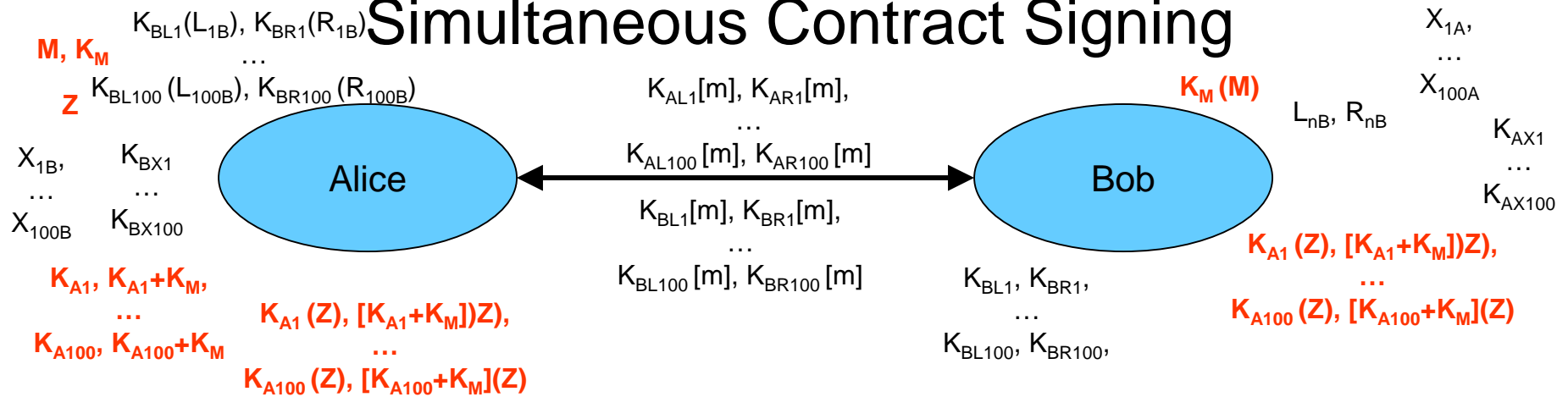
- Bob encrypts his receipts and sends them to Alice

Digital Certified Mail - Differences from Simultaneous Contract Signing



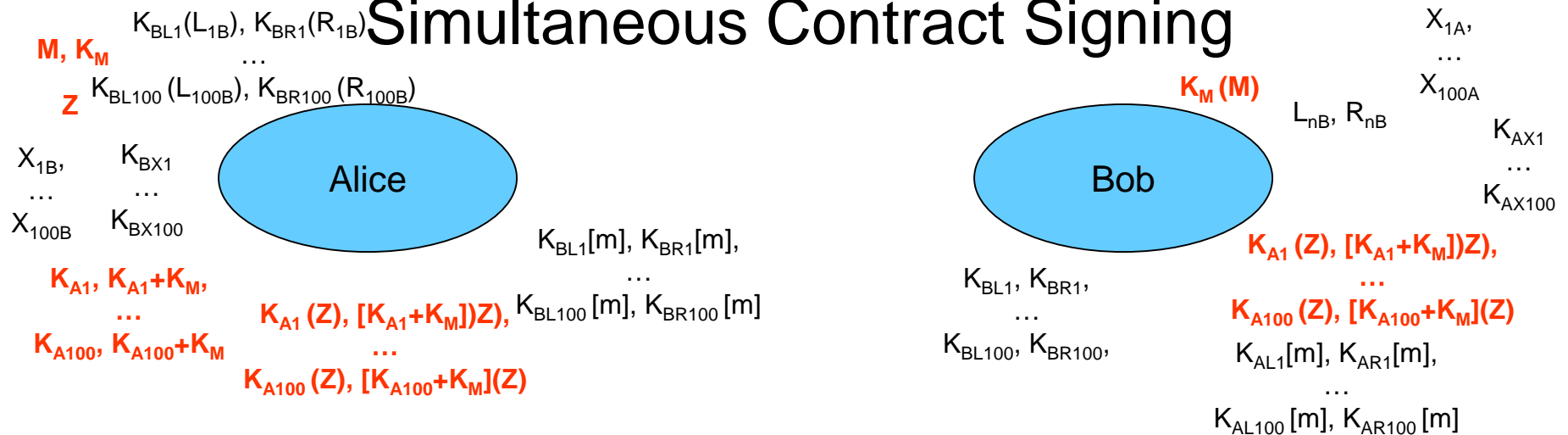
- Bob and Alice exchange their keys using oblivious transfer, as before

Digital Certified Mail - Differences from Simultaneous Contract Signing



- As before, Alice and Bob send each bit of their 200 keys (Note: this protocol certifies the mail, but does not protect its confidentiality, this exchange would have to be encrypted if that were a requirement)

Digital Certified Mail - Differences from Simultaneous Contract Signing

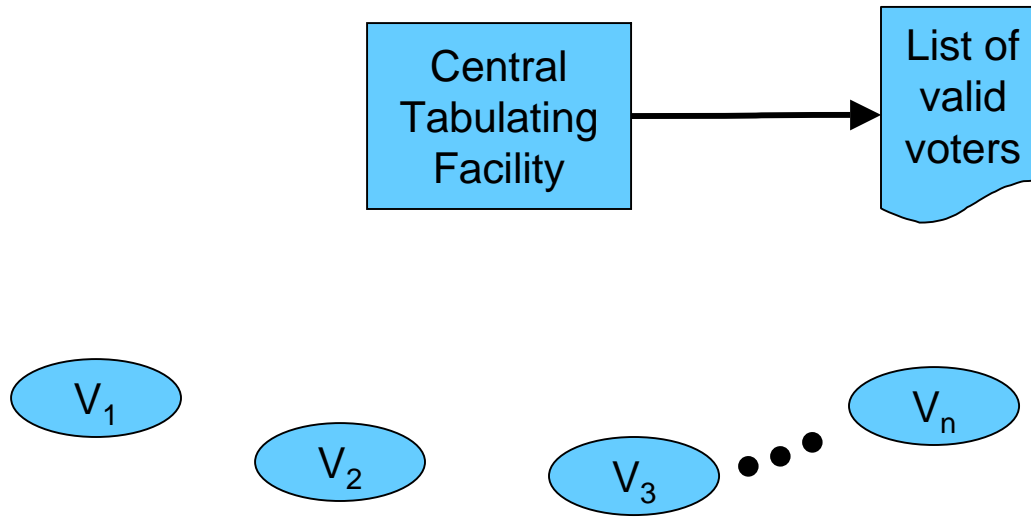


- Alice can use any right/left half set to validate Bob's receipt
- Bob can use any right/left half set to recover the message key
- The dummy messages generated by Alice are used to validate her oblivious transfer exchange

Secure Voting

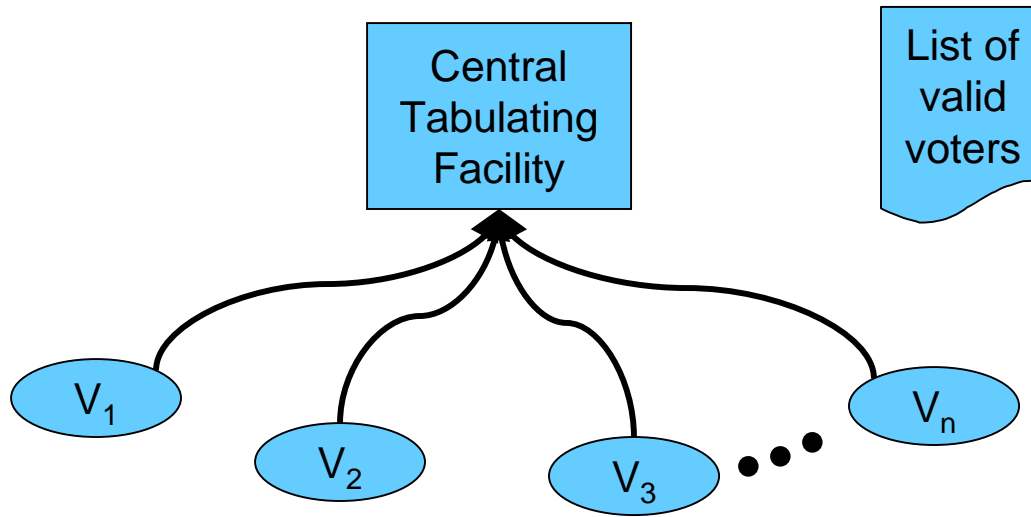
- Requirements for Secure Voting:
 - Only authorized voters are able to vote
 - No more than one vote per voter
 - Each voter's choice is kept confidential
 - Vote modification is detected
 - Voters can trace their vote to the final tally
- Additional option:
 - Voters/Abstainers are identifiable

Secure Voting



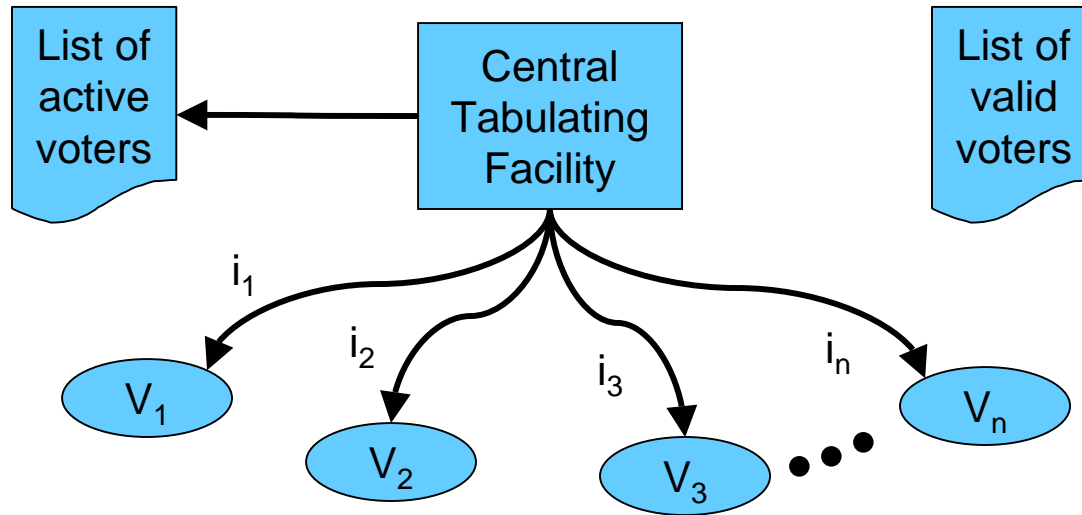
- CTF publishes a list of authorized voters

Secure Voting



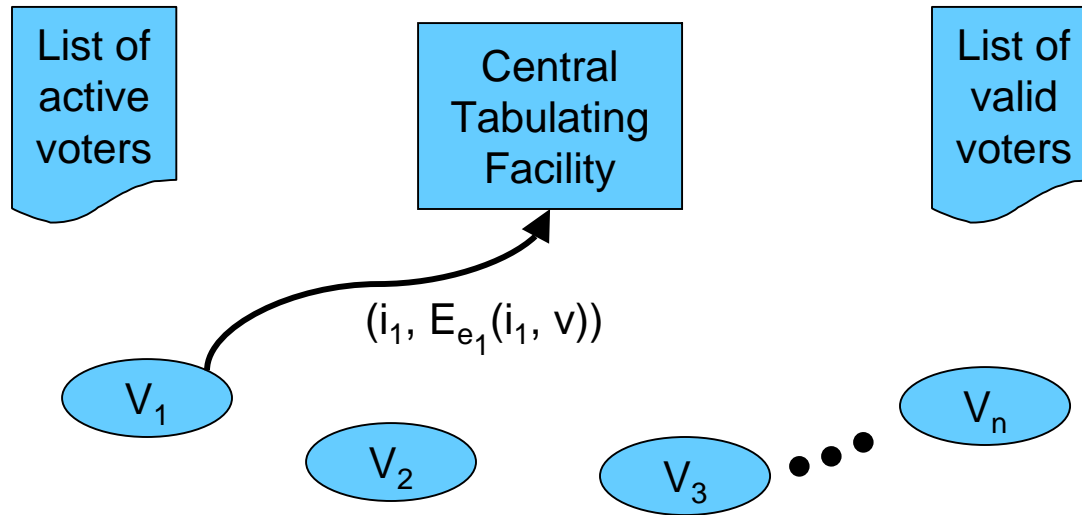
- Voters report their intentions to CTF

Secure Voting



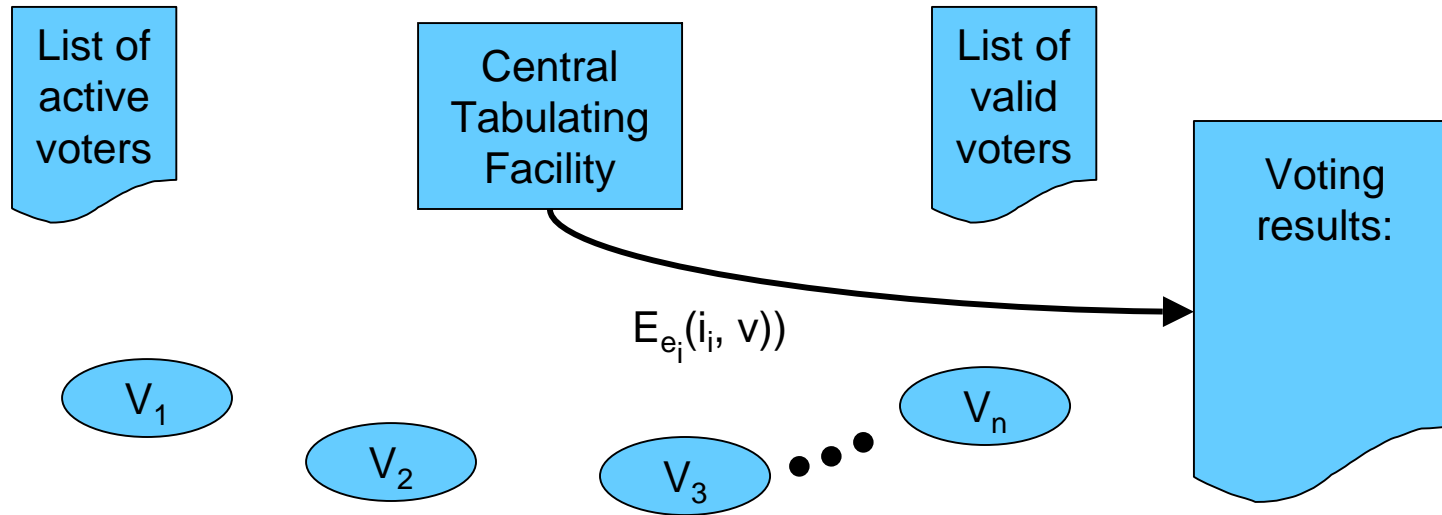
- CTF publishes list of participating voters and sends each voter an identification number, i , using an “All or Nothing Disclosure of Secrets” Protocol*

Secure Voting



- For each voter V_j :
 - Voter generates a public/private key pair, e_j, d_j
 - For vote, v , Voter generates a string and send it to CTF:
 $(i_j, E_{e_j}(i_j, v))$

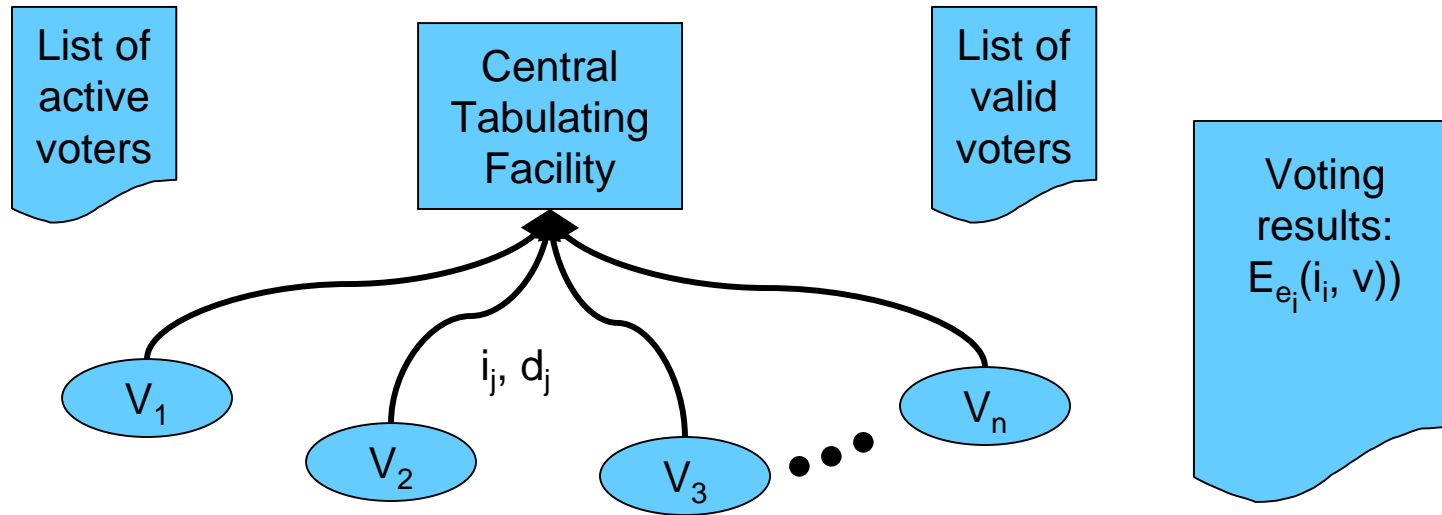
Secure Voting



- The CTF acknowledges the vote by publishing

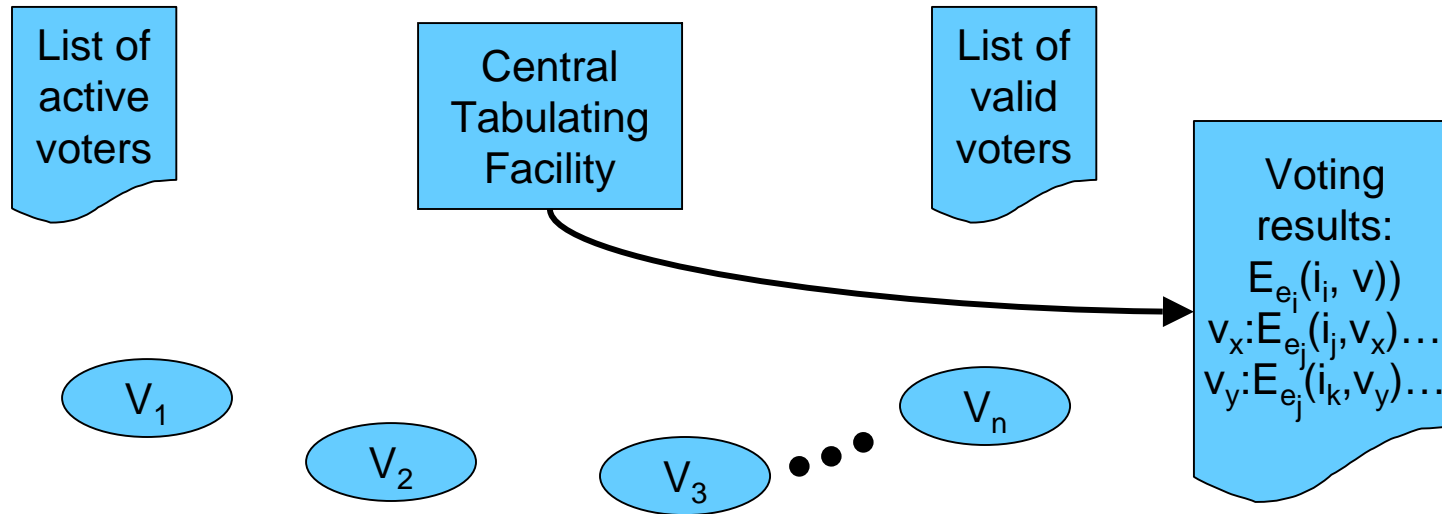
$$E_{e_1}(i_1, v)$$

Secure Voting



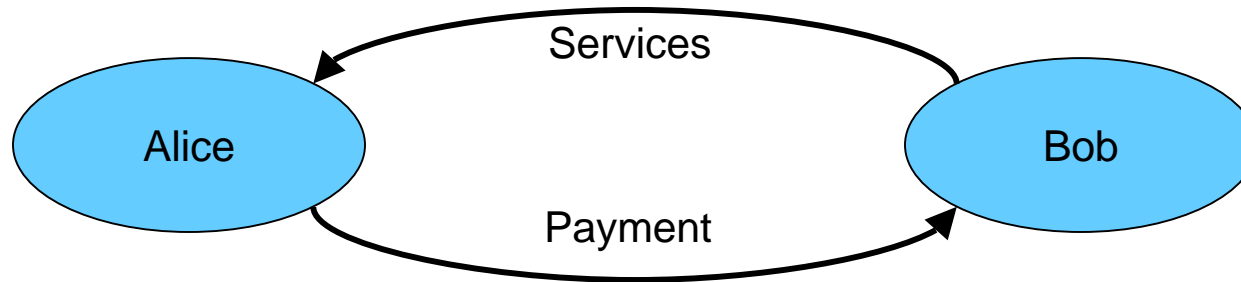
- Voters send the CTF i_j, d_j

Secure Voting



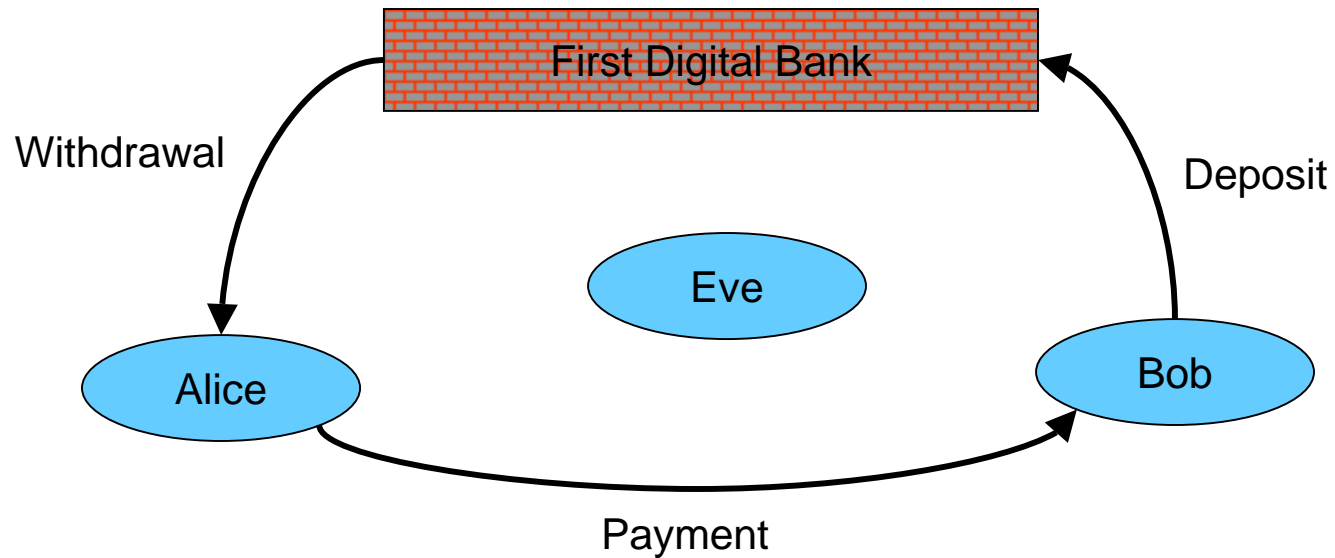
- CTF uses the d keys to decrypt messages and tally votes
- CTF publishes results and encrypted results for each vote

Digital Cash



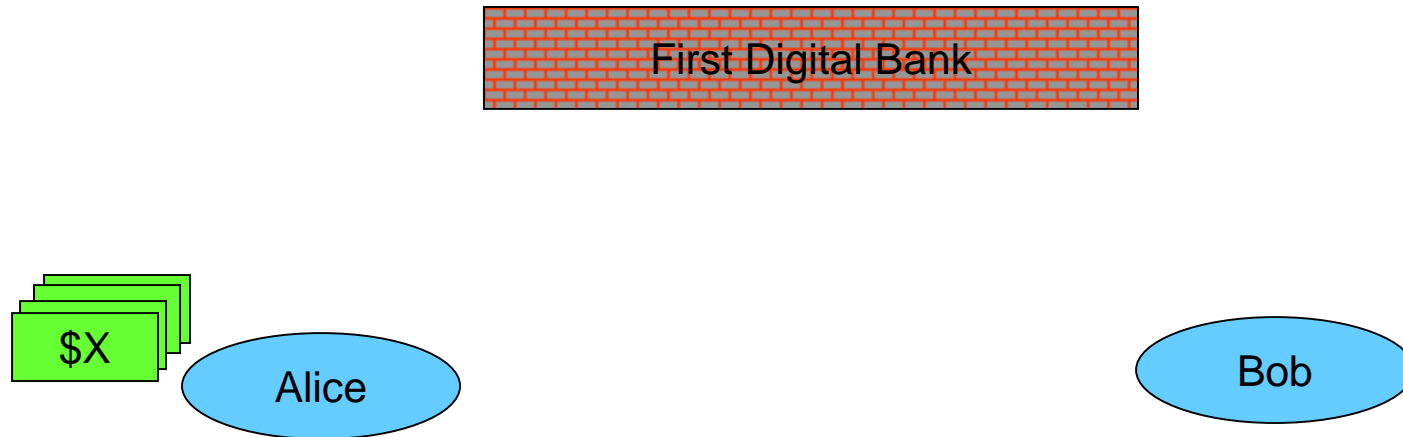
- Bob provides services to Alice, Alice pays Bob
- Alice could give Bob currency, but it is hard to move around and could be stolen
- Credit cards and checks leave an audit trail and can be recalled (particularly troublesome if Bob is a hit-man)
- Bob and Alice are customers for digital cash

Digital Cash



- Basic requirements for digital cash
 - Alice can only spend her digital cash tokens once
 - Bob can deposit or spend his digital cash tokens once
 - The bank, receiving a deposit from Bob, does not know where it came from
 - Eve can watch transactions, but cannot link Bob's deposit to Alice's payment

Digital Cash

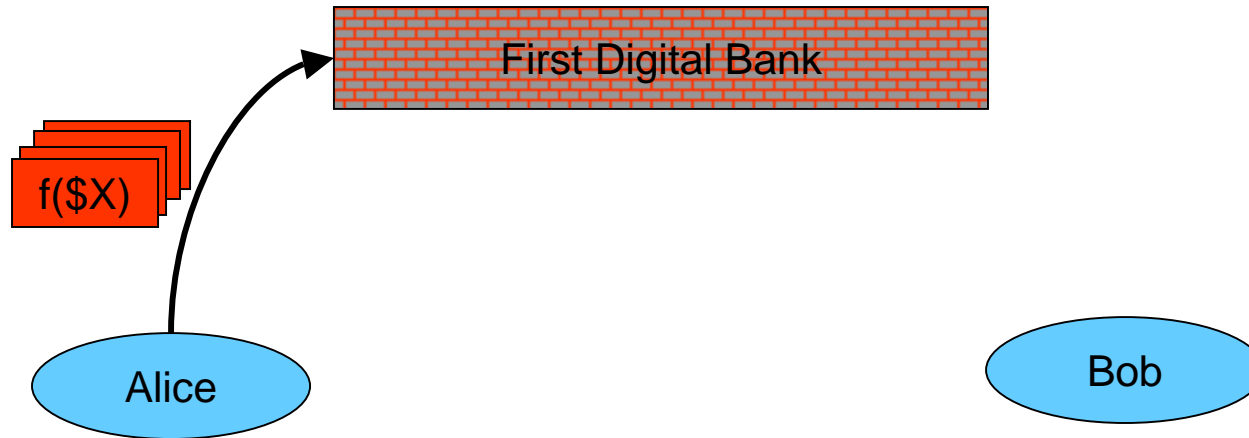


- Alice prepares N anonymous money orders for $\$X$. Each money order is formatted as:

Amount: $\$X$
Nonce: R_i
Identity Strings: $I_1=(I_{1L}, I_{1R})$
 $I_2=(I_{2L}, I_{2R})$
 \dots
 $I_N=(I_{NL}, I_{NR})$

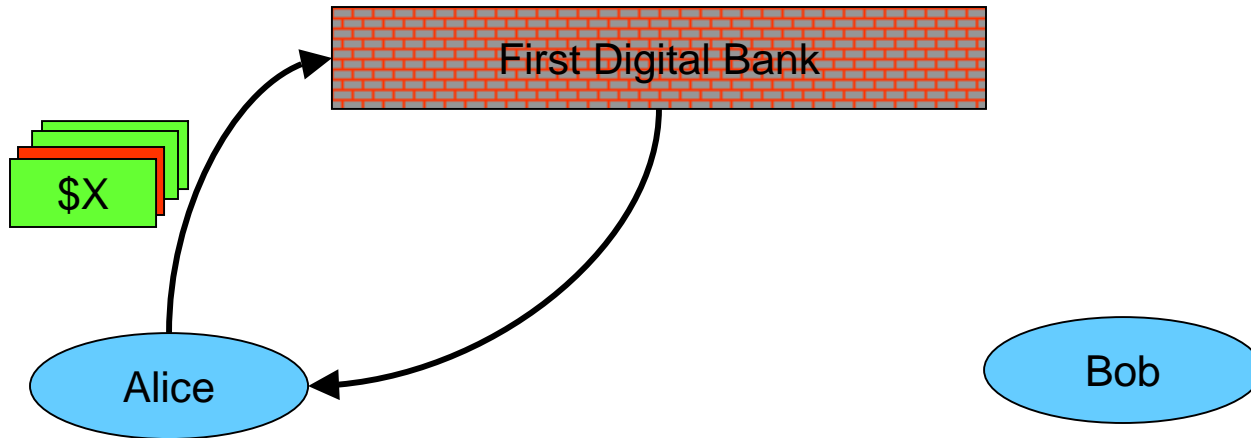
Each Identity String contains Alice's identifying information. The two halves of the identity string are generated by a secret splitting scheme (e.g., encrypt with a one-time pad. The cipher is I_L and the pad is I_R . Alice uses a bit commitment scheme to commit to each half.

Digital Cash



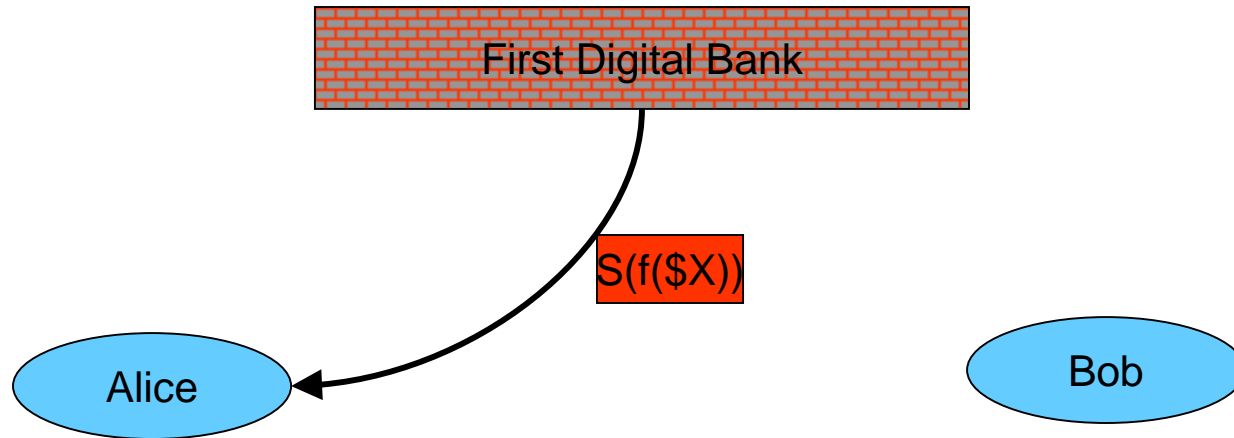
- Alice “blinds” the N money orders for a blind signature protocol (e.g., by multiplying by random numbers) and sends them to the bank

Digital Cash



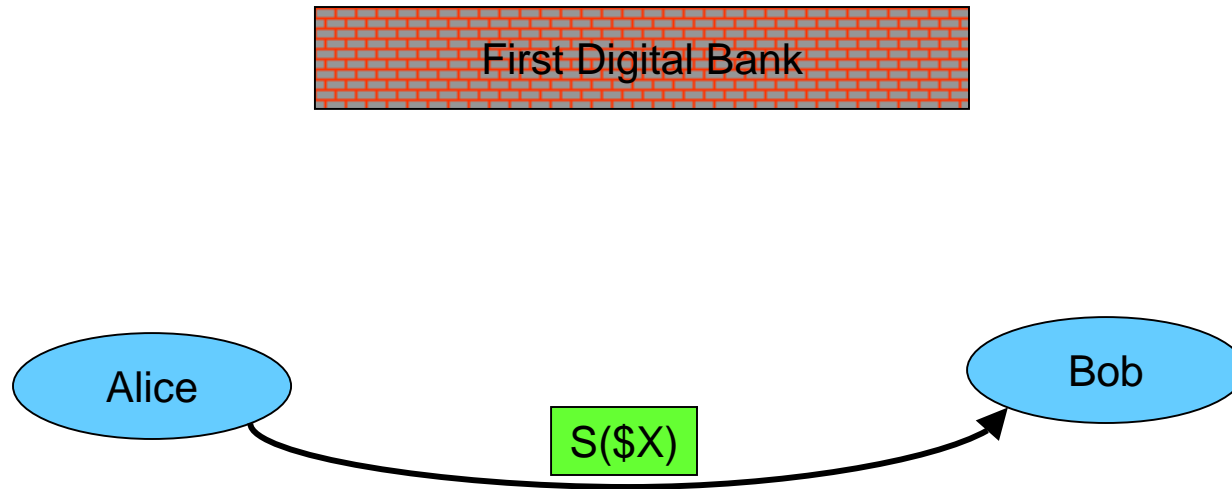
- The bank asks Alice to “unblind” a randomly chosen subset of $N-1$ of the money orders (e.g., money orders are unblinded by dividing by the blinding factor). This verifies that Alice has created the money orders properly (with probability $(N-1)/N$)

Digital Cash



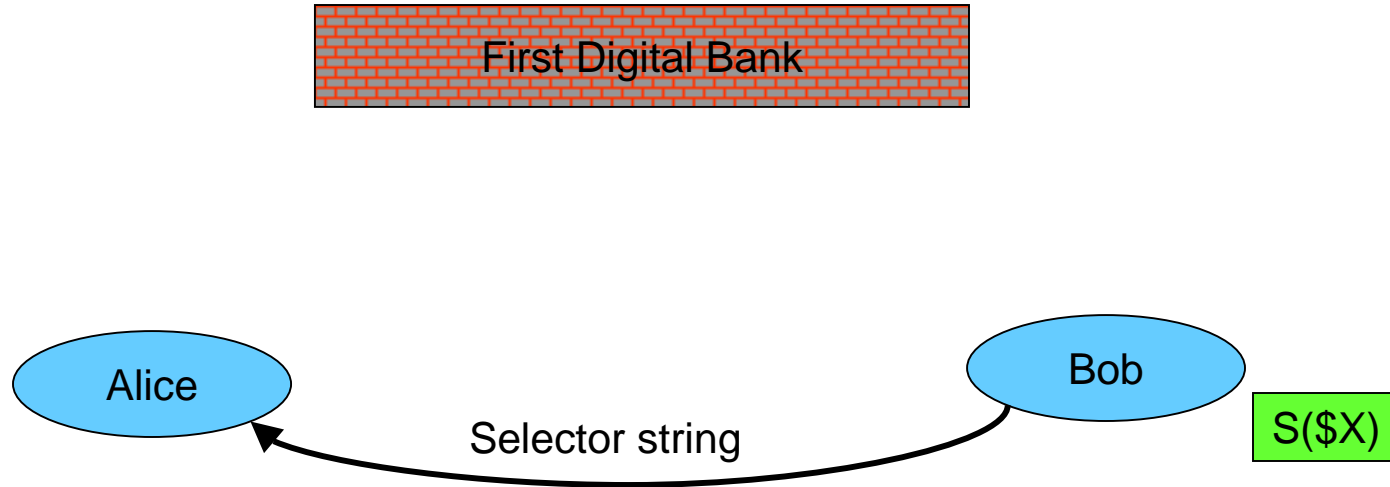
- Satisfied that Alice formed the other $N-1$ money orders properly, the bank signs the remaining money order, returns it to Alice and deducts $\$X$ from her account.

Digital Cash



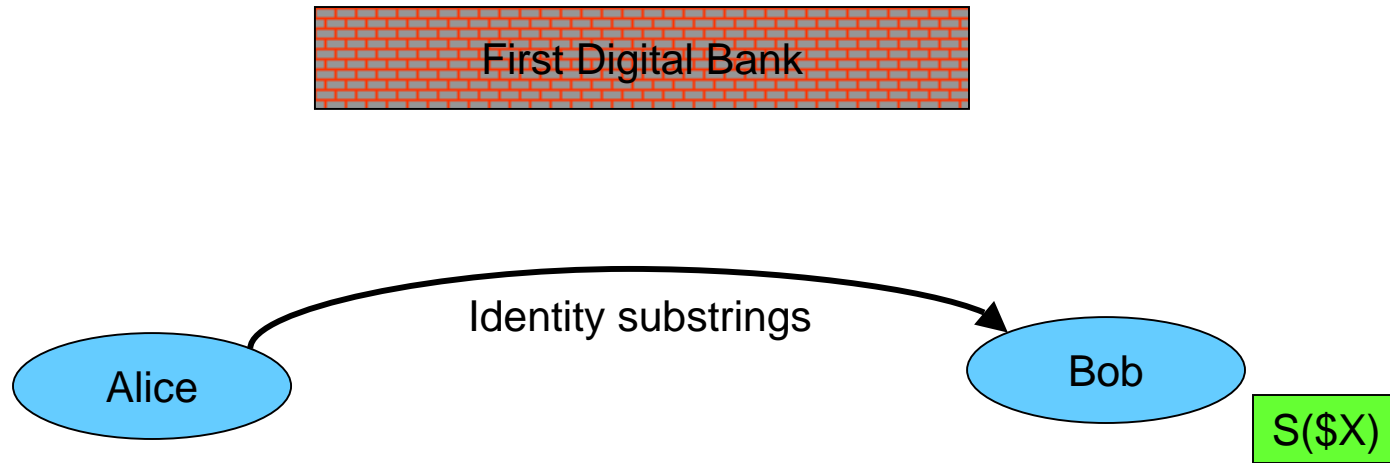
- Alice unblinds the signed money order and gives it to Bob as payment

Digital Cash



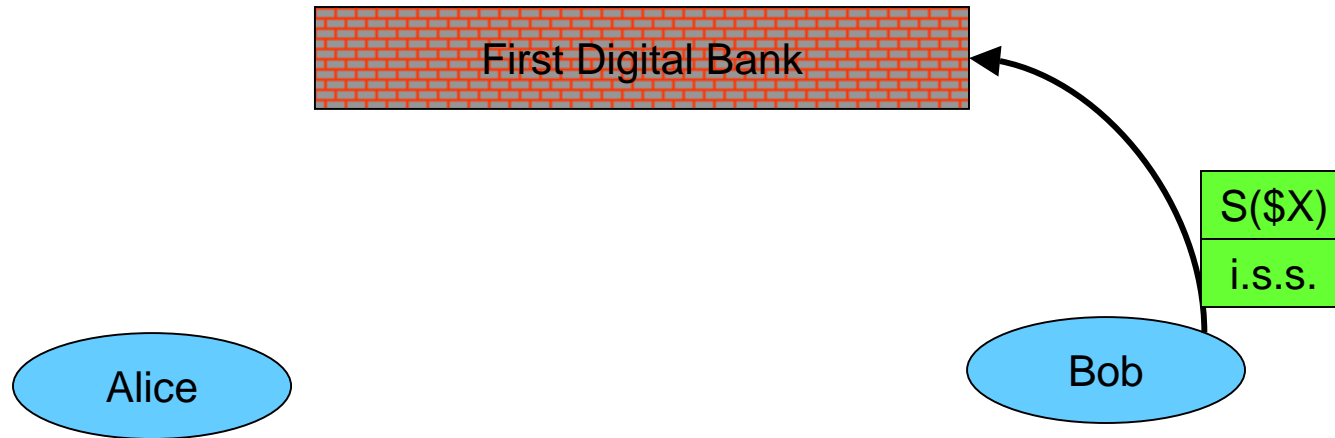
- Bob verifies the bank's signature
- Bob generates an N bit random number (selector string) and uses it to ask Alice to reveal either the right half or the left half of the Identity Strings
 - While knowledge of I_{jL} and I_{jR} would reveal Alice's identity, knowledge of I_{jL} and I_{kR} for different j and k does not.

Digital Cash



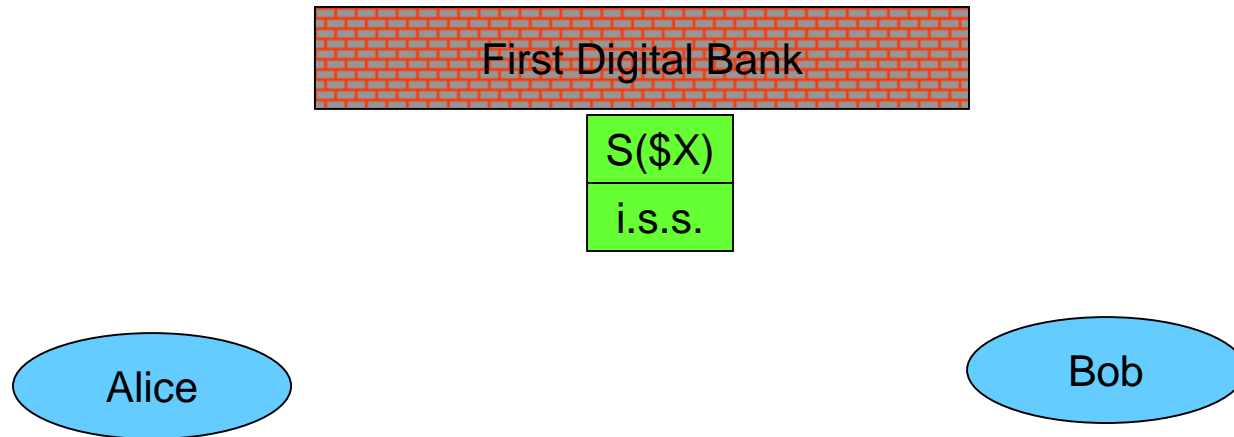
- Alice gives Bob the N halves of her Identity Strings

Digital Cash



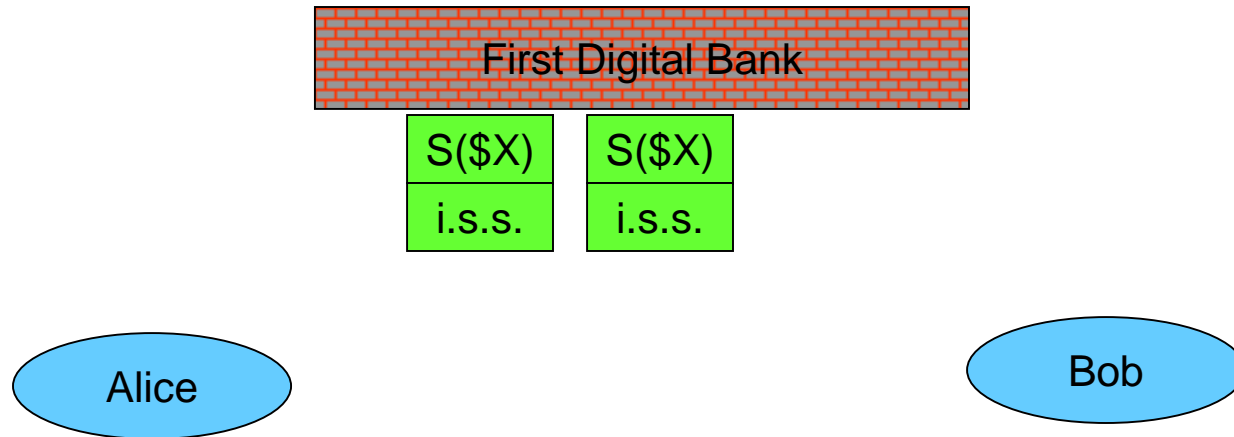
- Bob takes the money order to the bank

Digital Cash



- The bank verifies its signature and checks a database to see that a money order with the same uniqueness strings hasn't been spent.
- If there is no fraud, the bank deposits \$X into Bob's account and stores the uniqueness strings and the identity information in its database

Digital Cash



- If the bank finds a duplication, it matches the Identity Strings. It is very likely that at least one of the N identity substrings has revealed a left and right half for Alice. If so, Alice is found to be cheating.

Cheating with this Digital Cash Protocol

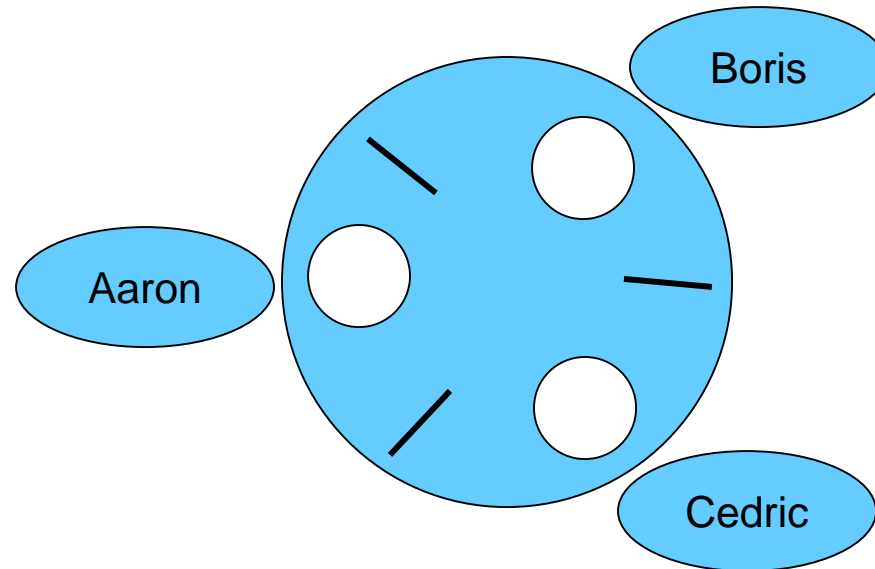
- Alice can copy a money order, she can't spend the copy without revealing her identity and the fact that she cheated.
- Alice can't modify the money order to hide her identity - doing so invalidates the bank's signature
- Alice doesn't need to be present when the money order is validated
- Alice could try to fool the bank with a bogus identity string in the first place. Since the bank examines $N-1$ money orders, her chances of cheating are $1/N$
- Bob can't cheat - if he tried copying the money order, the selector strings would match, revealing his action
- Alice and Bob can't cheat the bank through collusion: the bank has signed only one money order and will not accept a second with the same uniqueness string. Either Alice or Bob will be implicated in the attempt
- The bank cannot learn Alice's identity, even with Bob's assistance - its signature was blinded by Alice and Alice's identity is hidden between the two halves of her Identity String

Digital Cash

- Properties for an ideal digital cash system:
 - Independence - Unlike physical cash, the security of the digital cash is not dependent on any physical location; the cash can be transmitted through a network
 - Security - Digital cash cannot be copied and reused
 - Privacy (untraceability) - No one can link the user to their purchases
 - Off-line payment - Real-time connectivity between the user, merchant and bank is not needed to complete the transaction
 - Transferability - Digital cash can be given to other users
 - Divisibility - A piece of digital cash can be subdivided into smaller pieces

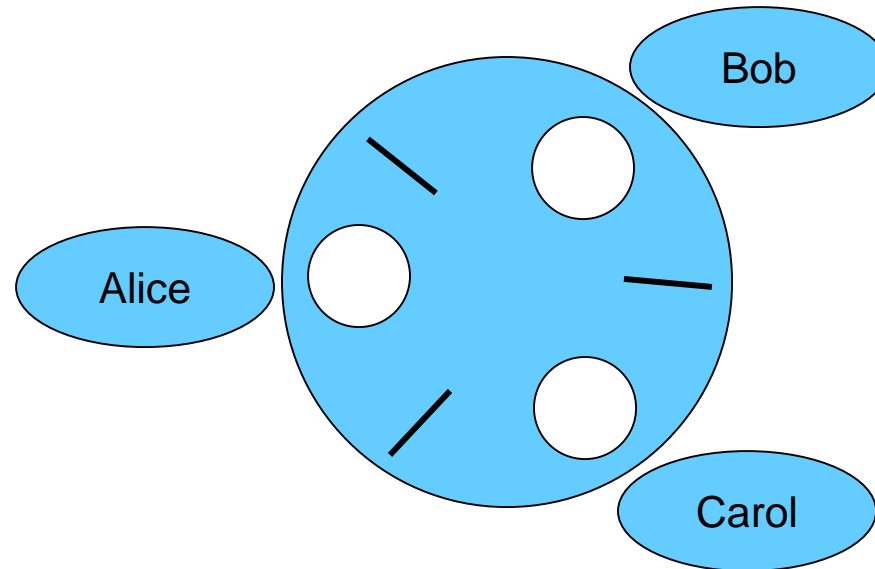
Anonymous Message Broadcast

First, an Aside



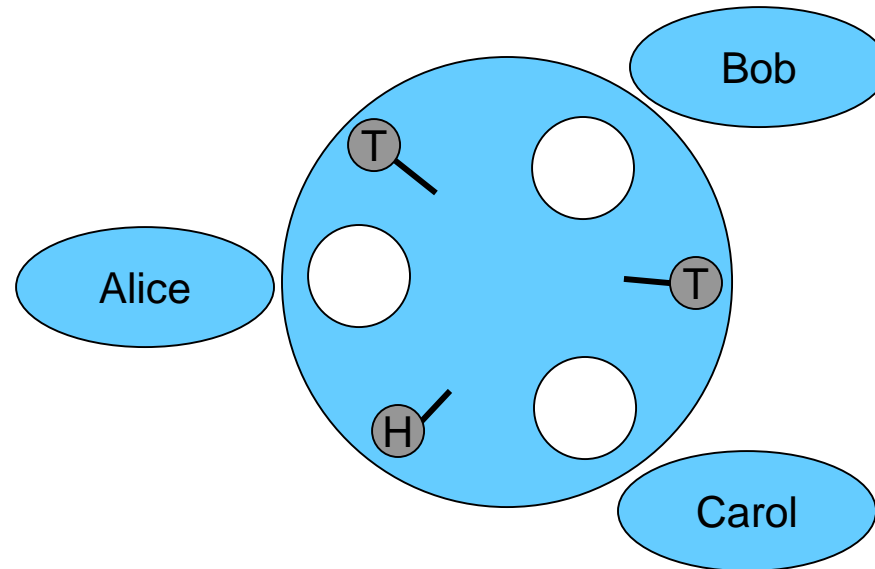
- Classic Computer Science Problem-
 - The Dining Philosopher's Problem:
 - Three philosophers are seated around a circular table. Each has a plate of food in front of them. Between each is a single chop-stick. Two chop-sticks are needed to eat. A philosopher does not put down their chop-stick(s) until they have finished eating.
 - How do you prevent a deadlock? - each taking one chop-stick: enough to prevent the others from eating, but not enough to allow any to complete

Anonymous Message Broadcast



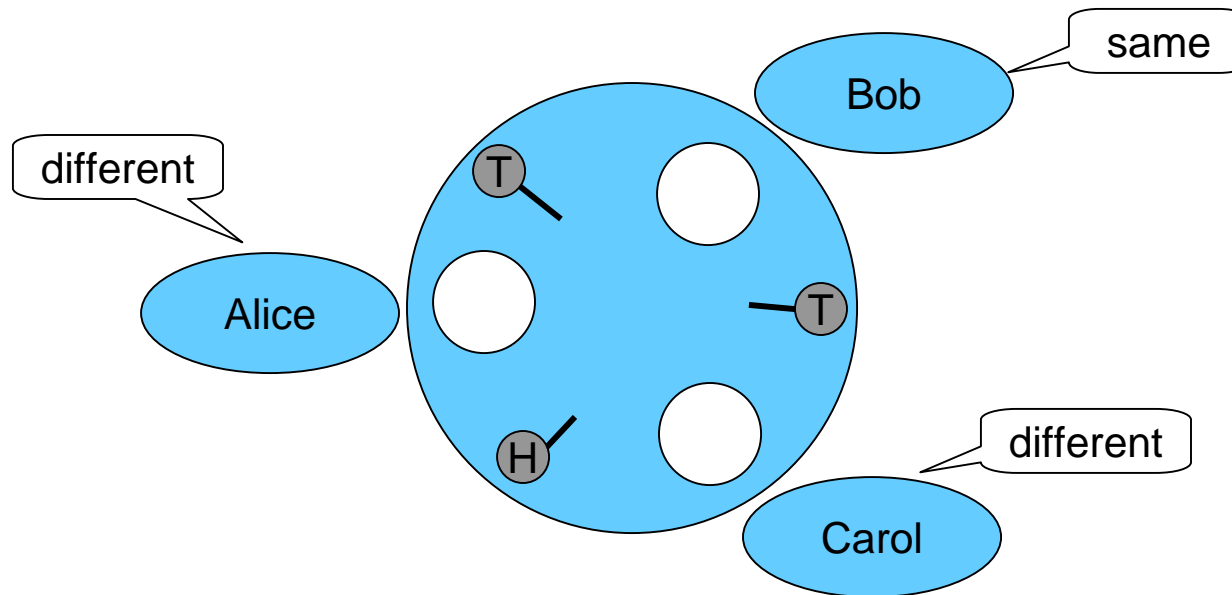
- A problem in security-
 - The Dining Cryptographer's Problem:
 - Three cryptographers are smart enough to solve the dining philosopher's problem, they design semaphores and a critical section to schedule the use of their chopsticks.
 - The waiter, Walter, tells them that the bill will be paid anonymously - either secretly by one of the cryptographers or by Norman, a representative of an unnamed agency from Maryland.
 - How do Alice, Bob, and Carol determine if (a) one of them is paying or (b) Norman is paying?

Anonymous Message Broadcast



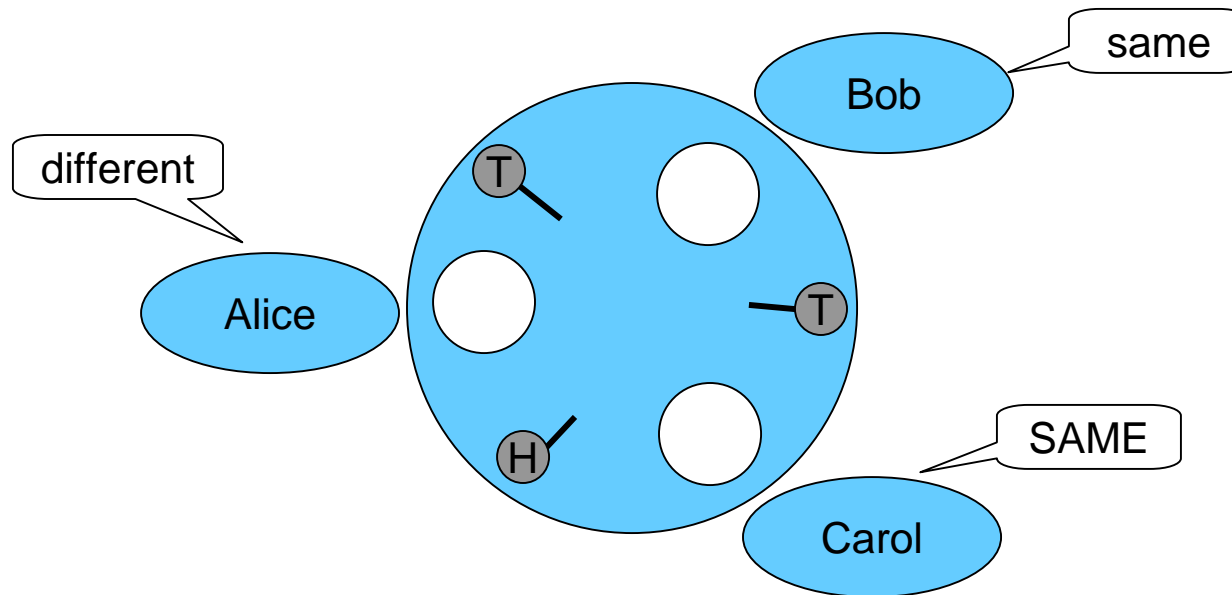
- A problem in security-
 - The Dining Cryptographer's Problem:
 - Alice, Bob and Carol flip 3 unbiased coins so that only the two adjacent parties can see the outcome

Anonymous Message Broadcast



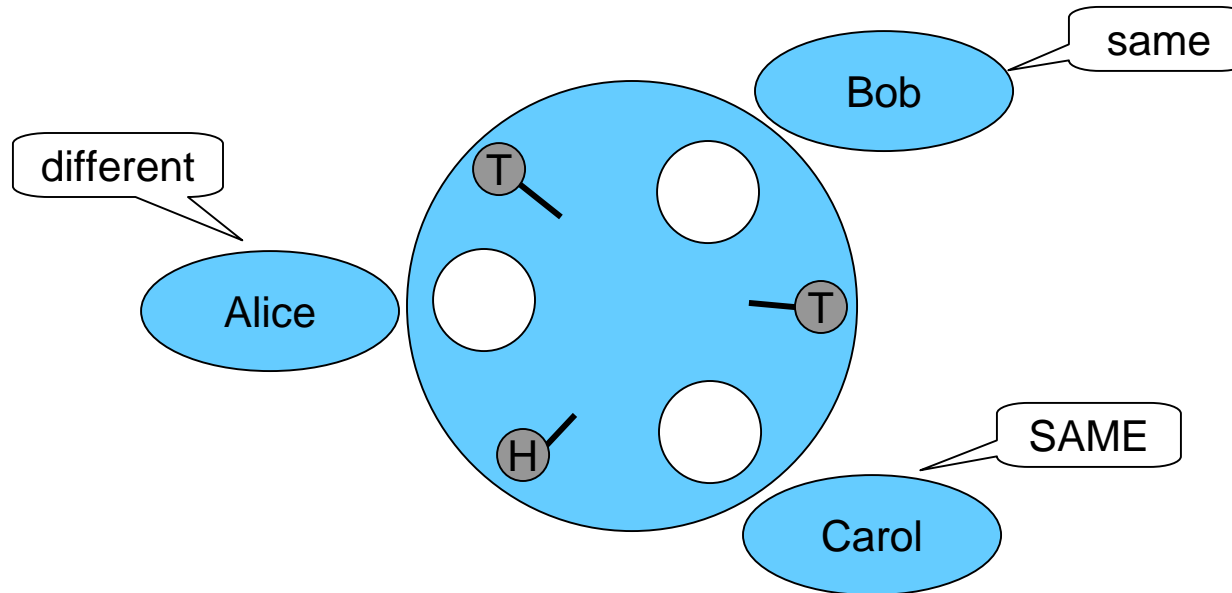
- A problem in security-
 - The Dining Cryptographer's Problem:
 - Alice, Bob and Carol flip 3 unbiased coins so that only the two adjacent parties can see the outcome
 - Each (except the payer) announces whether the two coins they see are the same or different
 - If an even number of differences indicates that the payer is Norman

Anonymous Message Broadcast



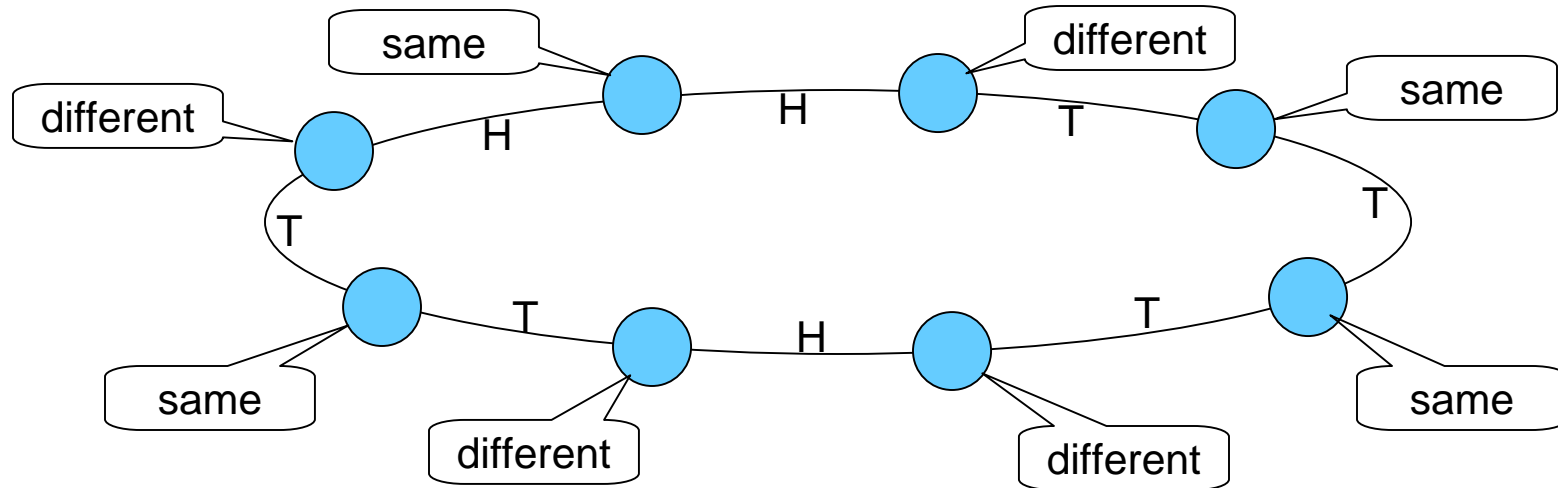
- A problem in security-
 - The Dining Cryptographer's Problem:
 - Alice, Bob and Carol flip 3 unbiased coins so that only the two adjacent parties can see the outcome
 - Each (except the payer) announces whether the two coins they see are the same or different
 - If an odd number of differences indicates that the payer is at the table

Anonymous Message Broadcast



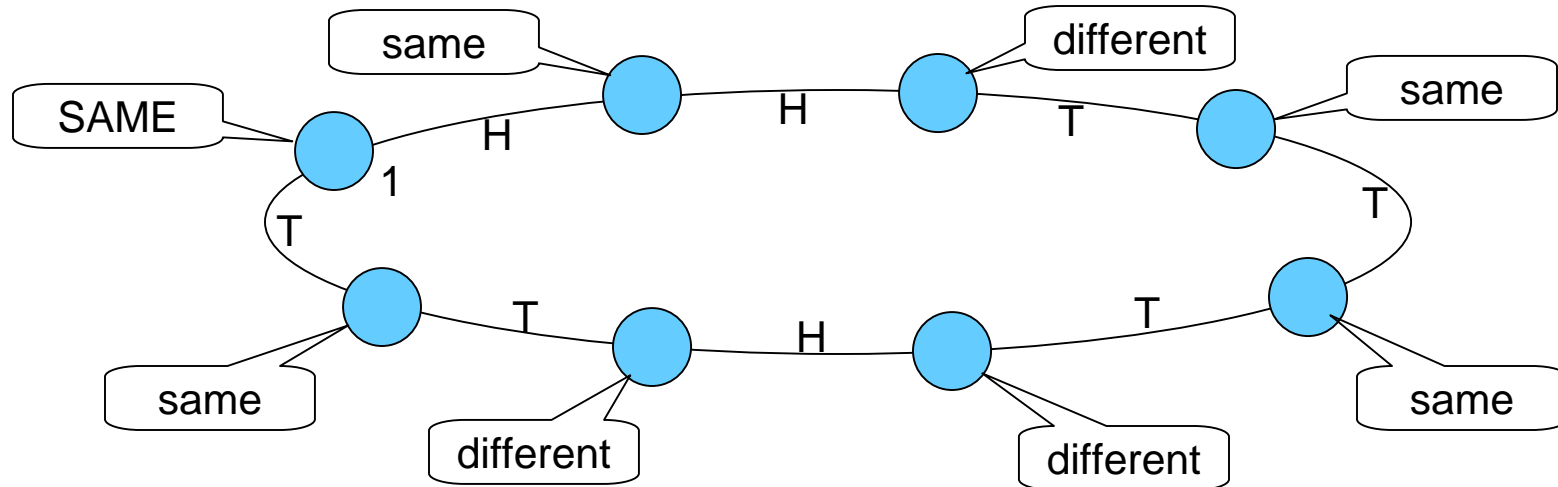
- A problem in security-
 - The Dining Cryptographer's Problem:
 - Alice, Bob and Carol flip 3 unbiased coins so that only the two adjacent parties can see the outcome
 - Each (except the payer) announces whether the two coins they see are the same or different
 - If an odd number of differences indicates that the payer is at the table
 - But the cryptographers don't know which one paid

Anonymous Message Broadcast: Unconditional Sender and Recipient Untraceability



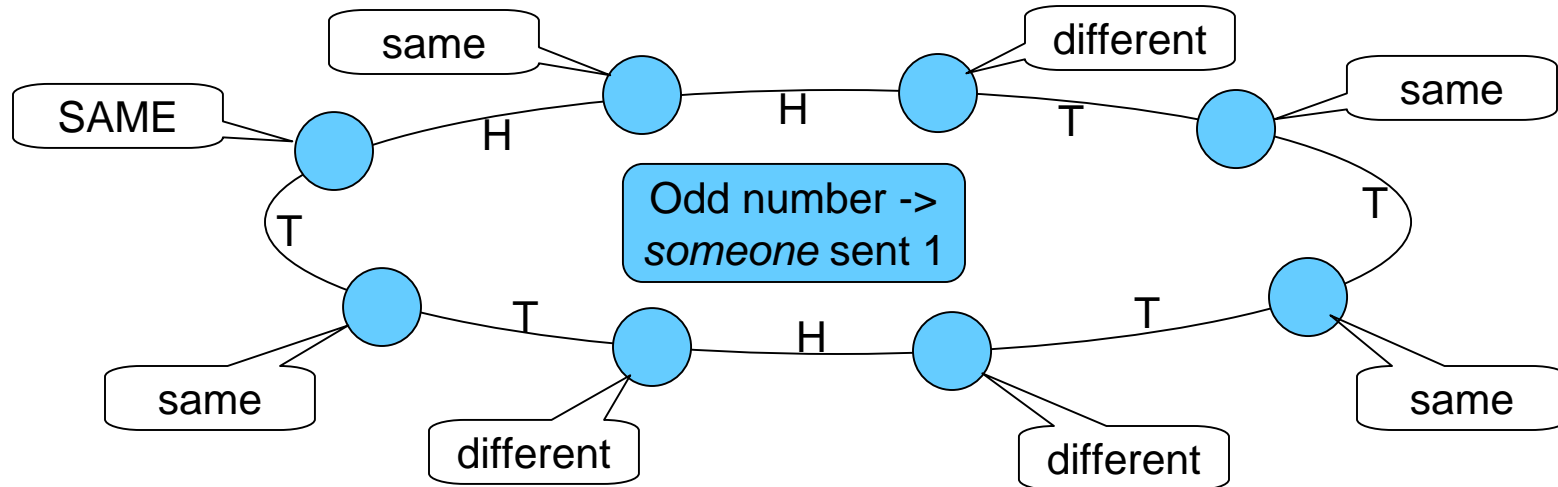
- Nodes arrange themselves in a logical circle
- Periodically, adjacent pairs of nodes randomly (and secretly) “flip a coin” using an encrypted fair coin flip protocol.
- Periodically, each node announces “same” or “different” - the number of “different” messages should be even

Anonymous Message Broadcast: Unconditional Sender and Recipient Untraceability



- To broadcast a 1, a node inverts the “same”/“different” message
- If nodes see that the outcome of the protocol differs from the bit they sent, it means that someone else collided with them. They back off a random interval and try again (e.g., as an IEEE 802.3 network operates)

Anonymous Message Broadcast: Unconditional Sender and Recipient Untraceability



- As described, sender of message cannot be determined.
- If underlying message were first encrypted in public key of recipient, traffic analysis would be impossible

Security Protocols

