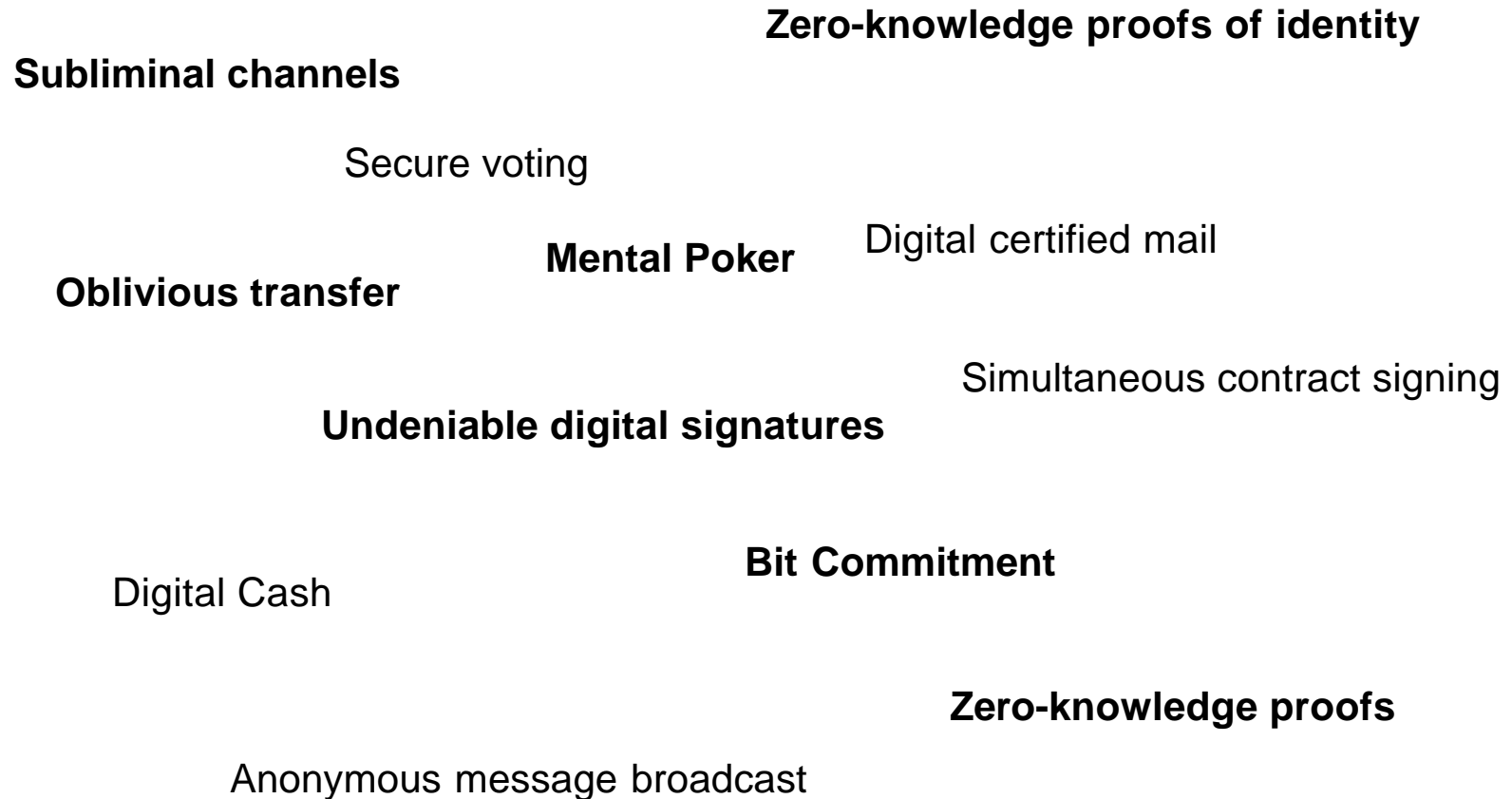


NIS/CpE 691CE

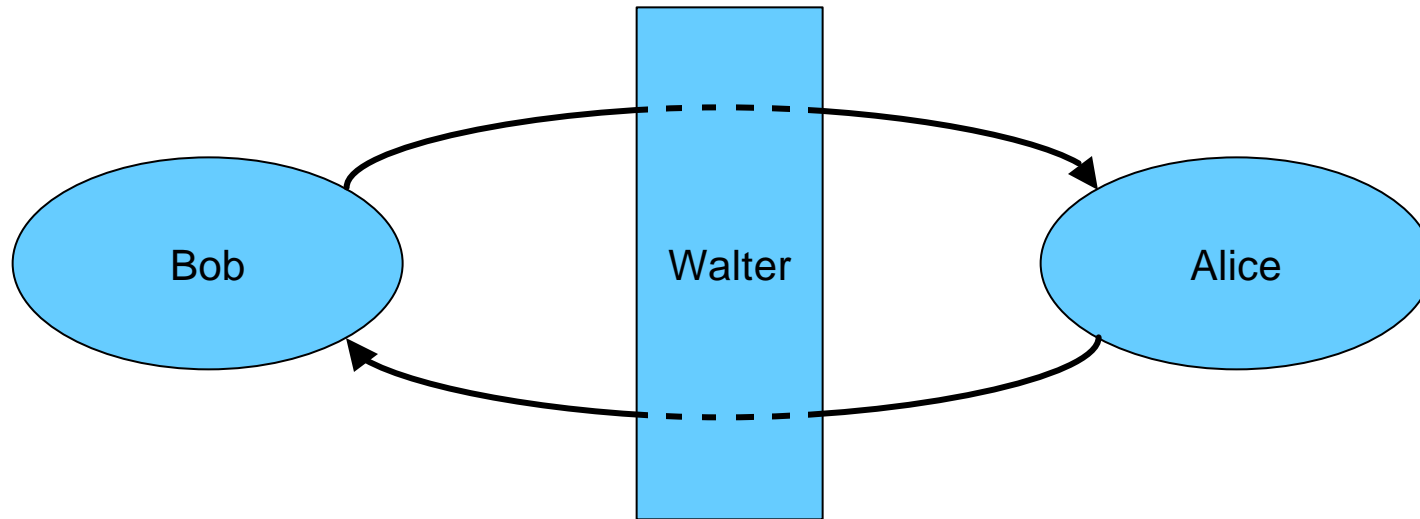
Information System Security

Class 10 – 11/25/02

Security Protocols

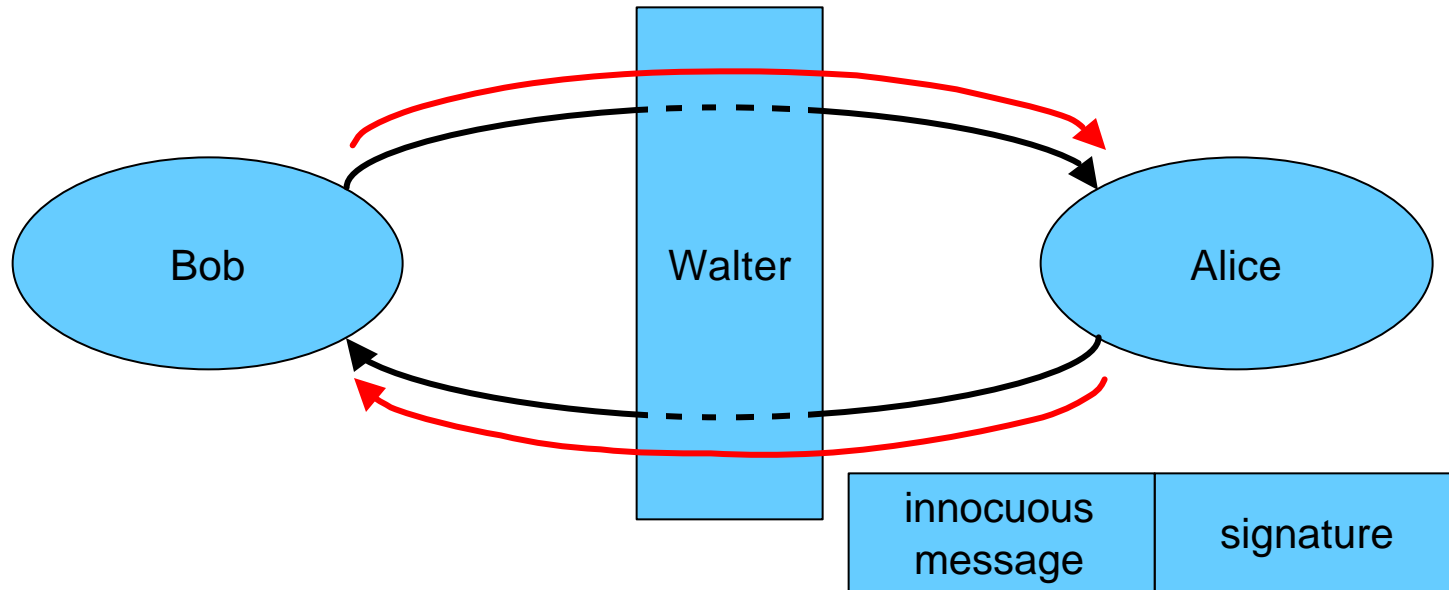


Subliminal Channels



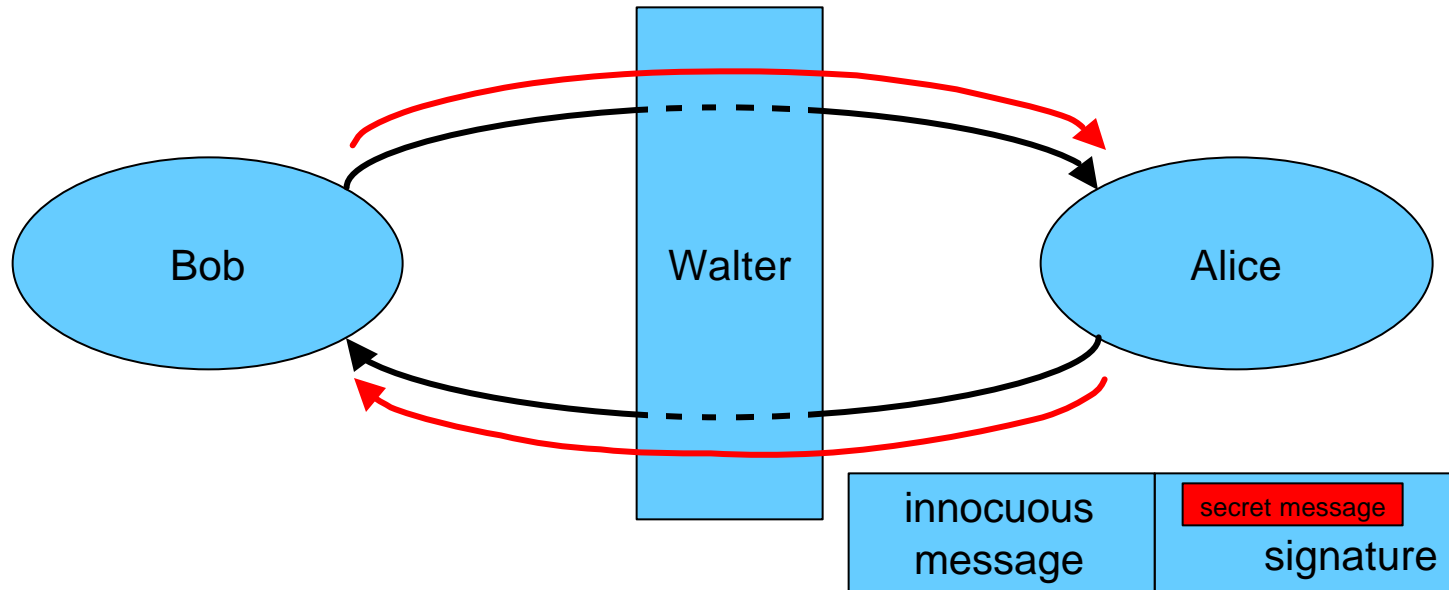
- Alice and Bob want to exchange secret messages, of necessity, through an untrusted intermediary, Walter
- Walter is willing to relay innocuous messages, but might modify them or insert bogus ones: he is not willing to let Alice and Bob exchange secret messages

Subliminal Channels



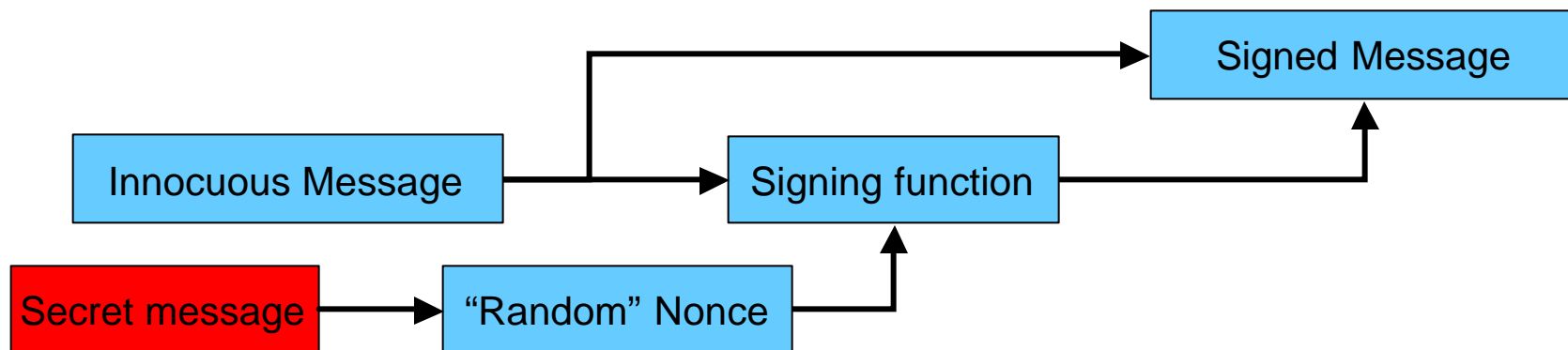
- Alice and Bob want to exchange secret messages, of necessity, through an untrusted intermediary, Walter
- Walter is willing to relay innocuous messages, but might modify them or insert bogus ones: he is not willing to let Alice and Bob exchange secret messages
- Alice and Bob digitally sign their messages to detect Walter's phony messages

Subliminal Channels



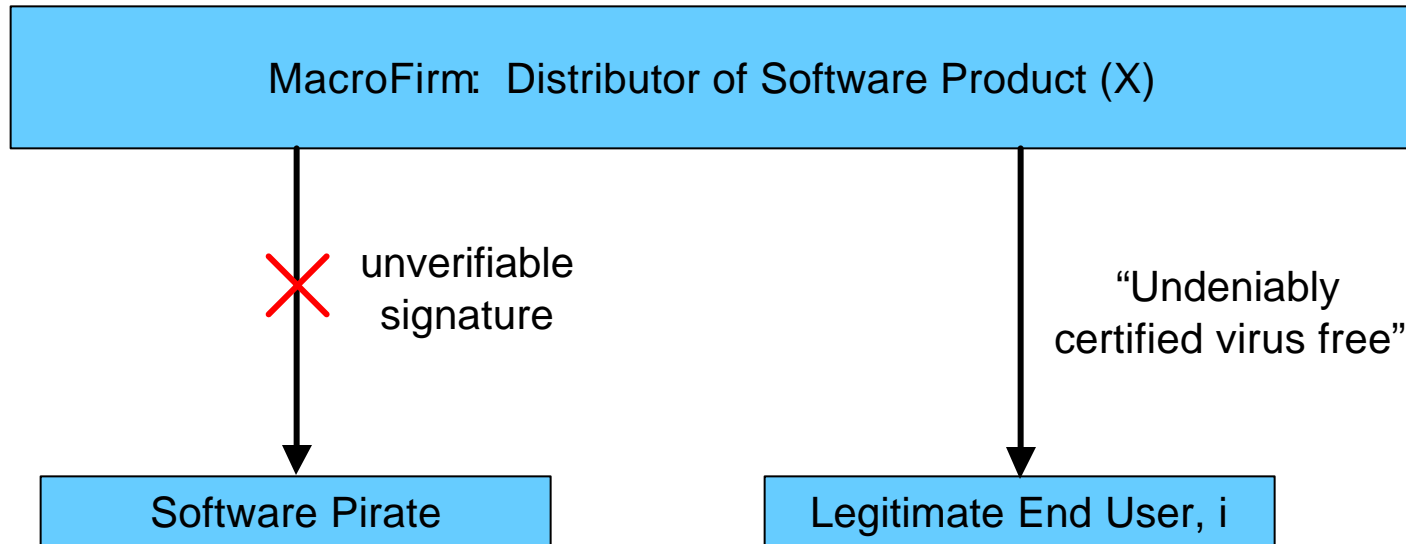
- Alice and Bob want to exchange secret messages, of necessity, through an untrusted intermediary, Walter
- Walter is willing to relay innocuous messages, but might modify them or insert bogus ones: he is not willing to let Alice and Bob exchange secret messages
- Alice and Bob digitally sign their messages to detect Walter's phony messages
- Alice and Bob innocuous messages contain no information: the message is in the signature

Implementing the Subliminal Channel



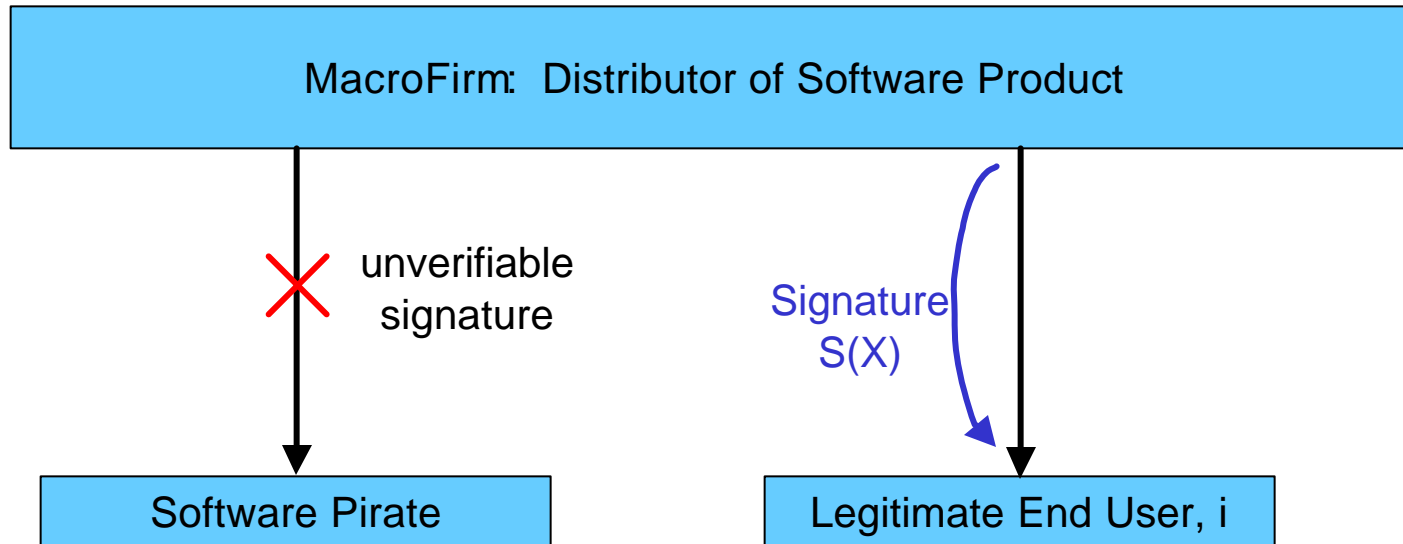
- Signature has properties resembling a normal signature
- As long as information rate of "secret" message is low, compared to innocuous message, detection can be minimized.

Undeniable Digital Signatures



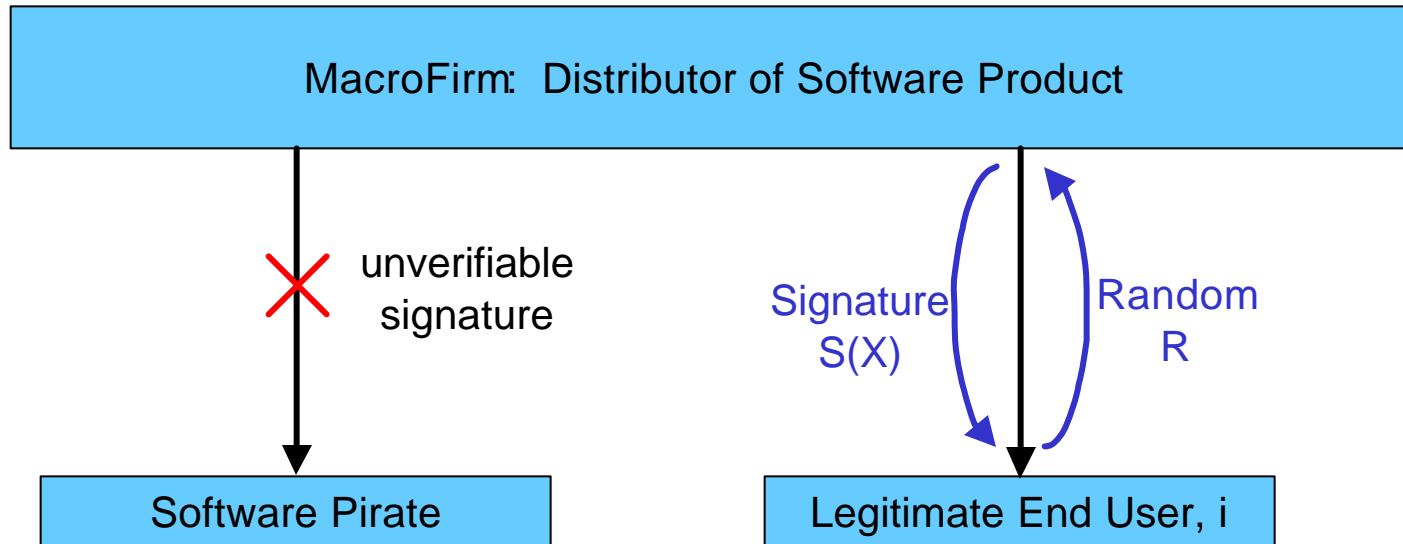
- MacroFirm is willing to verify their digitally signed software to valid end users, but not to users who have obtained software illegally
- The act of verifying the signature to a legitimate user should not provide a mechanism for a software pirate to distribute authenticated software

Implementing an Undeniable Digital Signatures



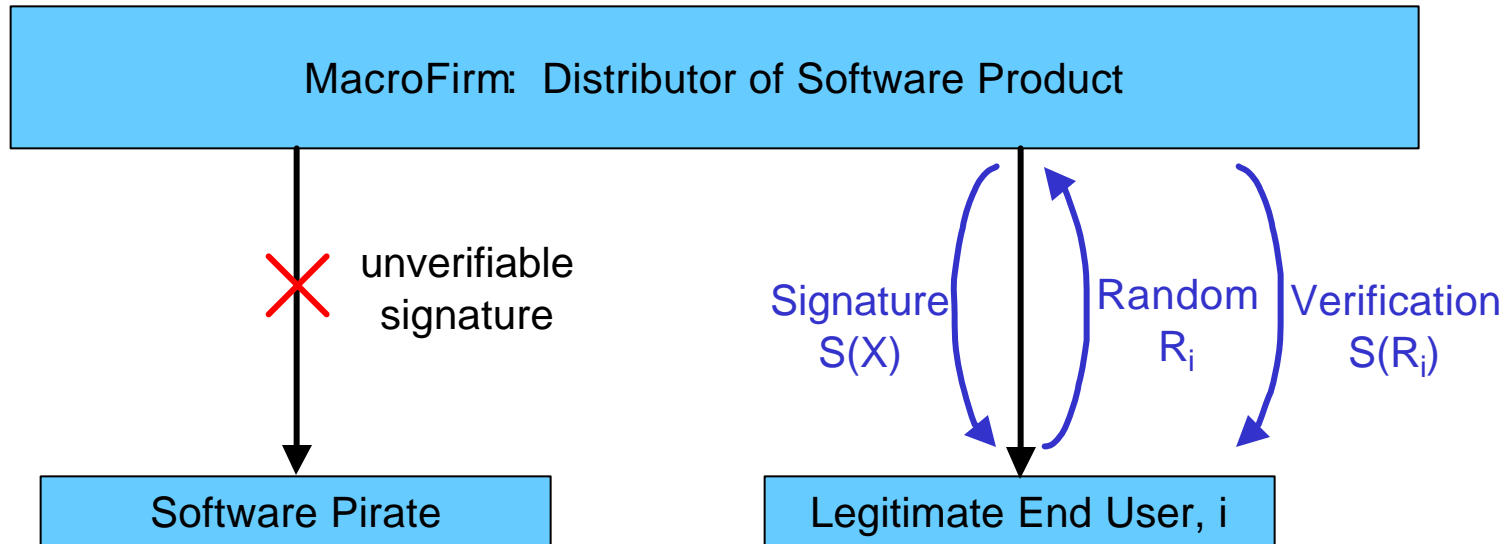
- MacroFirm is willing to verify their digitally signed software to valid end users, but not to users who have obtained software illegally
- The act of verifying the signature to a legitimate user should not provide a mechanism for a software pirate to distribute authenticated software

Implementing an Undeniable Digital Signatures



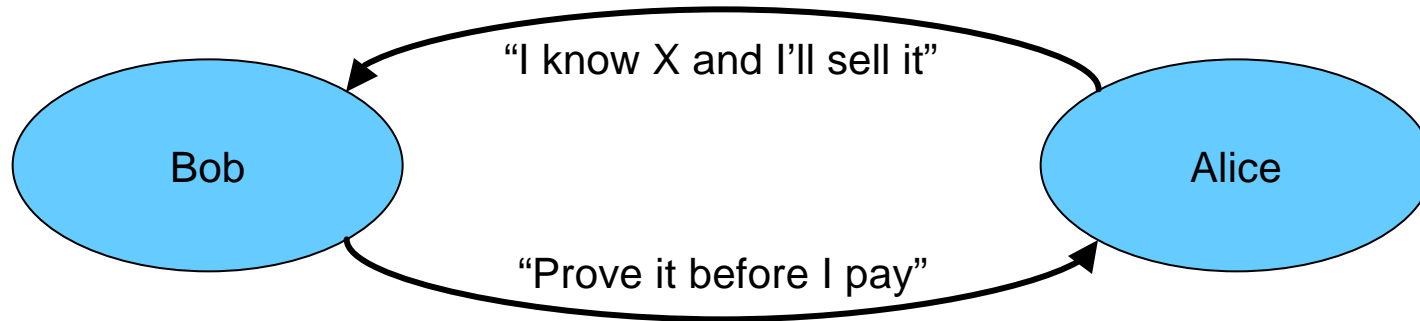
- MacroFirm is willing to verify their digitally signed software to valid end users, but not to users who have obtained software illegally
- The act of verifying the signature to a legitimate user should not provide a mechanism for a software pirate to distribute authenticated software

Implementing an Undeniable Digital Signatures



- MacroFirm is willing to verify their digitally signed software to valid end users, but not to users who have obtained software illegally
- The act of verifying the signature to a legitimate user should not provide a mechanism for a software pirate to distribute authenticated software
- Only MacroFirm knows the signature function $S()$ and could generate a signature
- Knowing R_i and $S(R_i)$ gives end user no advantage in generating $S(R_j)$ for another user

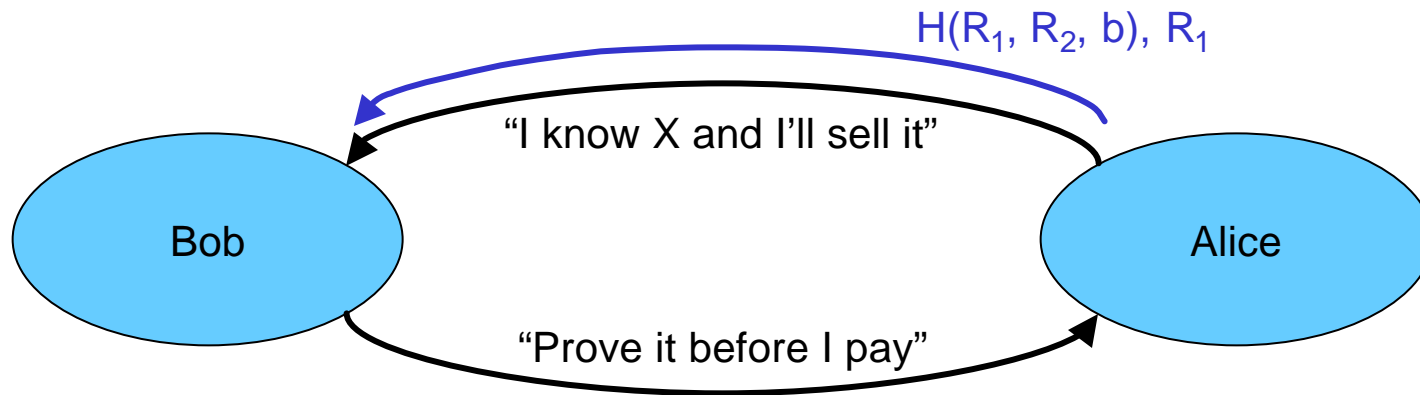
Bit Commitment



- Alice wants to sell a piece of information (X) to Bob
- Before Bob is willing to pay, he wants to be sure Alice knows X
- If Alice reveals X to Bob, Bob will then know X, why should he then pay?

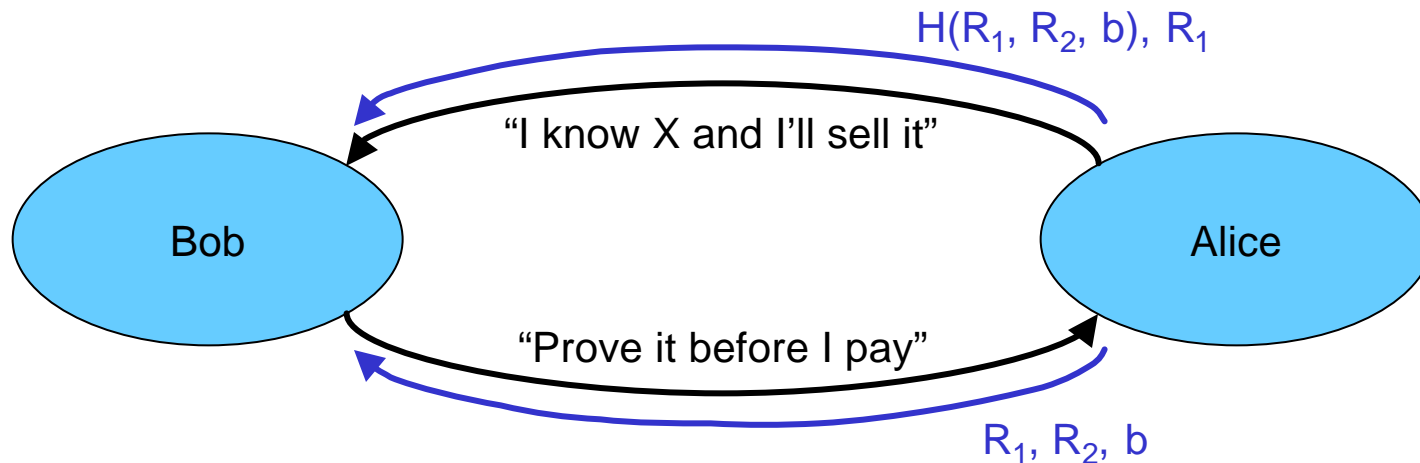
- Without loss of generality, assume that X is one bit, b

Implementation of a Bit Commitment Scheme



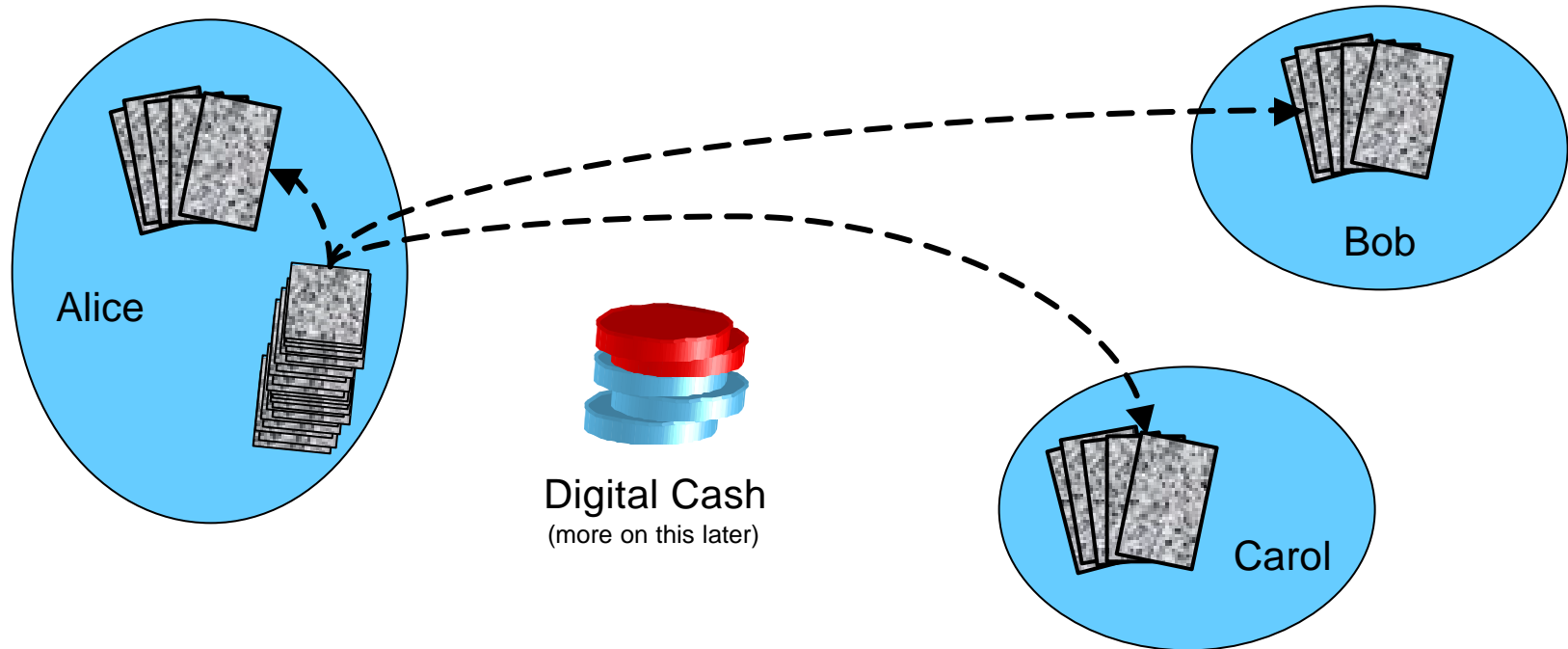
- Alice wants to sell a piece of information (X) to Bob
- Before Bob is willing to pay, he wants to be sure Alice knows X
- If Alice reveals X to Bob, Bob will then know X, why should he then pay?
- Without loss of generality, assume X is one bit, b
- Alice generates random bit strings, R_1 and R_2
- Alice generates a one-way hash: $H(R_1, R_2, b)$ and send it to Bob with R_1 , committing to b, but not revealing b

Implementation of a Bit Commitment Scheme



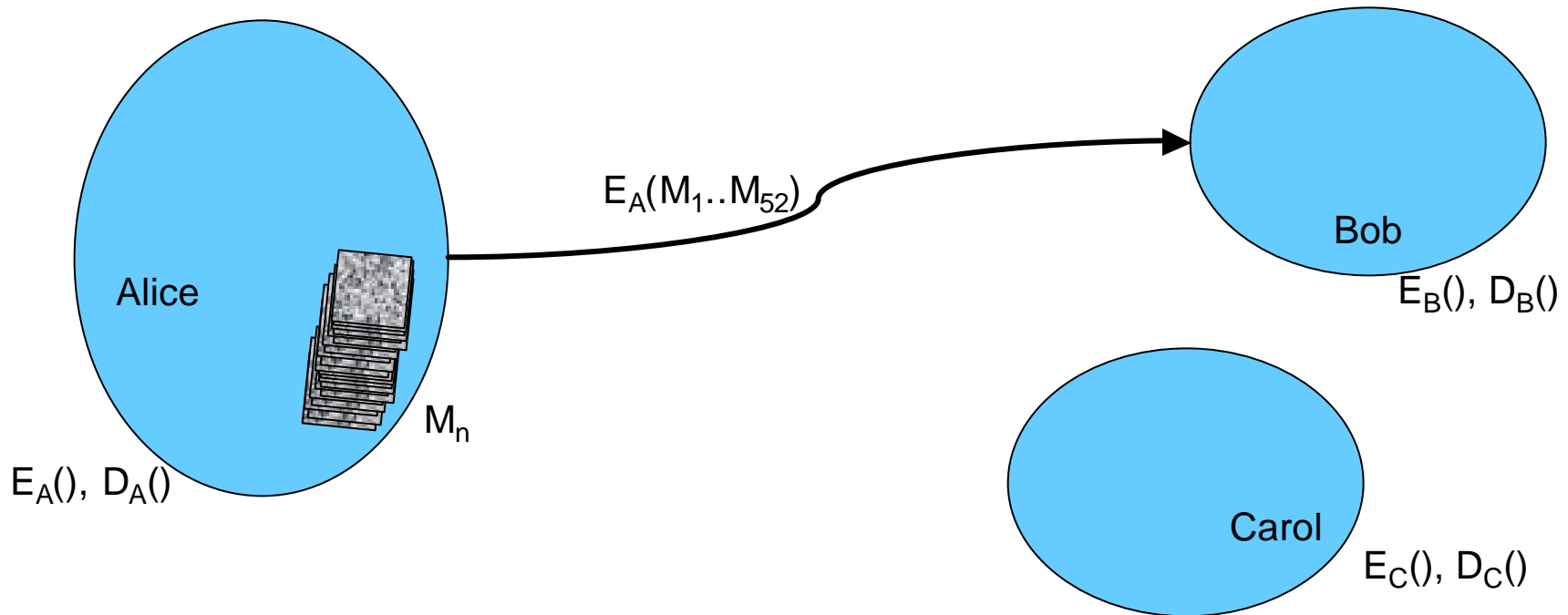
- Alice wants to sell a piece of information (X) to Bob
- Before Bob is willing to pay, he wants to be sure Alice knows X
- If Alice reveals X to Bob, Bob will then know X, why should he then pay?
- Without loss of generality, assume X is one bit, b
- Alice generates random bit strings, R_1 and R_2
- Alice generates a one-way hash: $H(R_1, R_2, b)$ and send it to Bob with R_1 , committing to b, but not revealing b
- Alice later sends Bob (R_1, R_2, b) , revealing b

Mental Poker



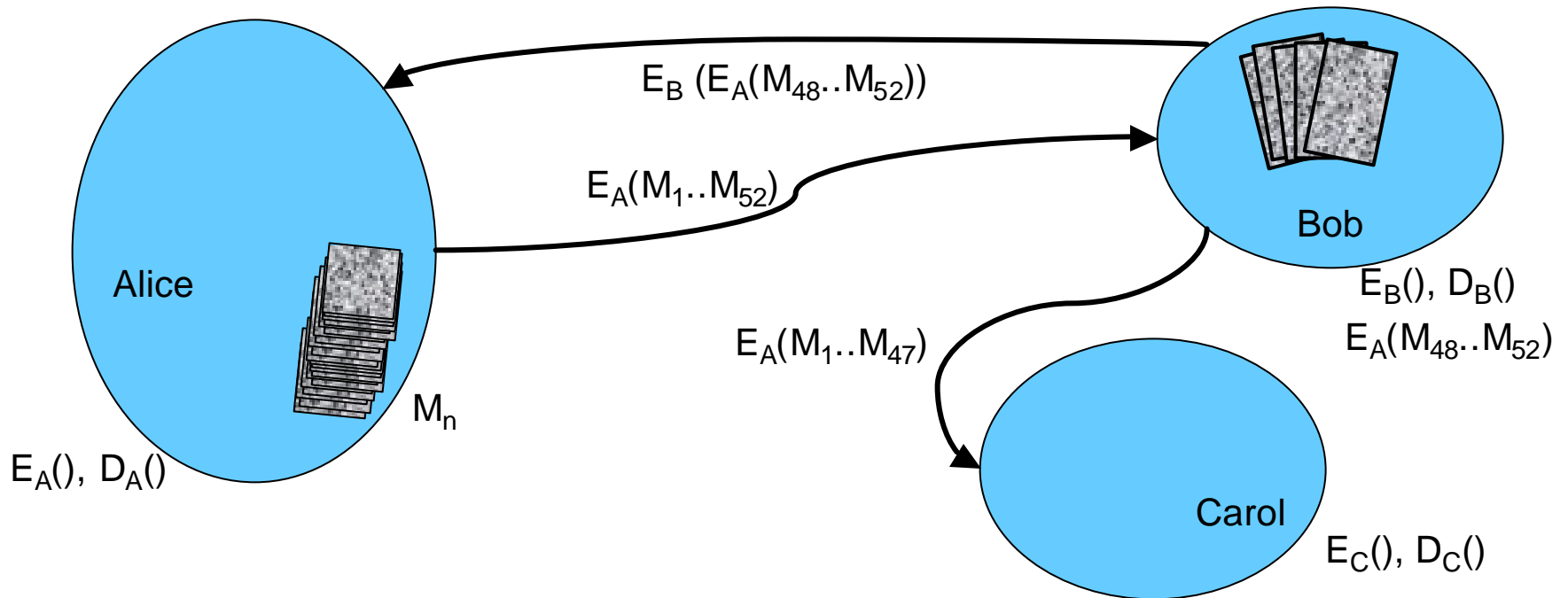
- Alice, Bob and Carol want to play Poker electronically (or mentally, if they have to cerebral capacity to perform logarithms and large number multiplications in their heads)
- How do they deal cards without the possibility of cheating (looking at each other's hands or lying about their own hands)?

Implementing Mental Poker



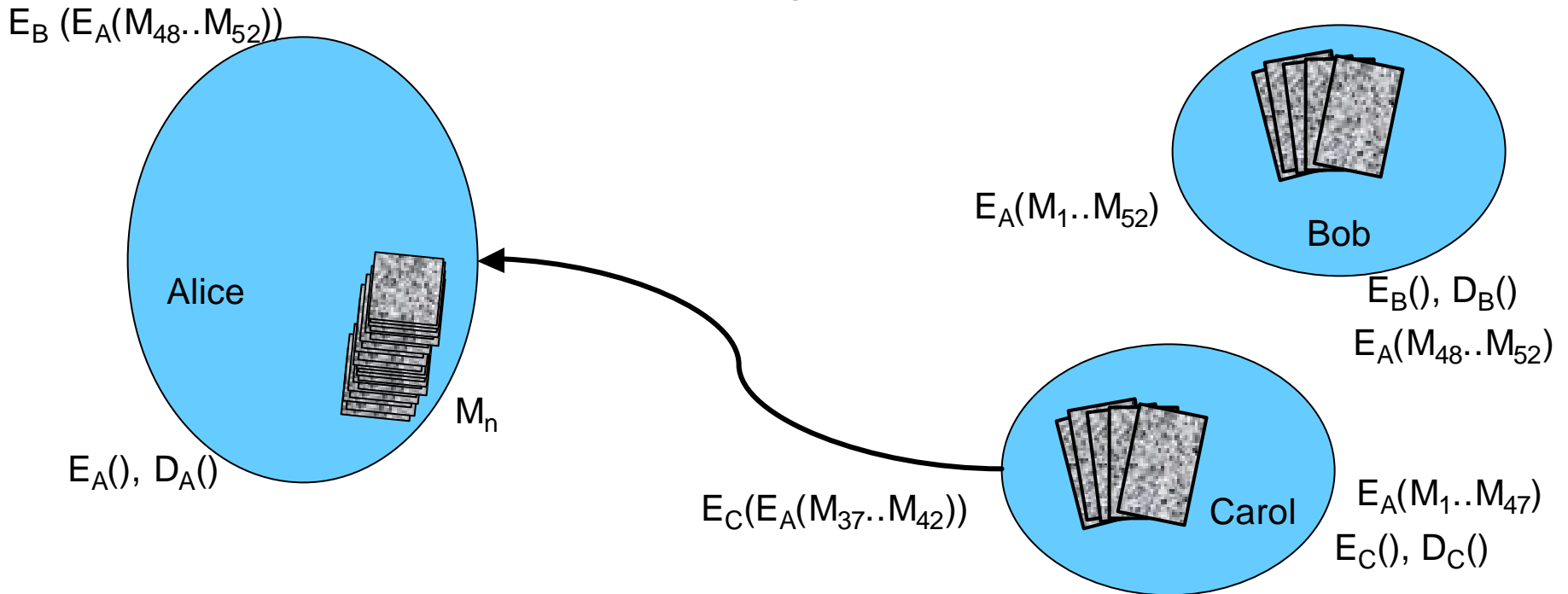
- Alice, Bob, and Carol all generate a secret key or a public/private key pair
- Alice (as dealer) generates 52 random numbers M_n with an authentication field
- Alice encrypts the 52 numbers and sends them to Bob

Implementing Mental Poker



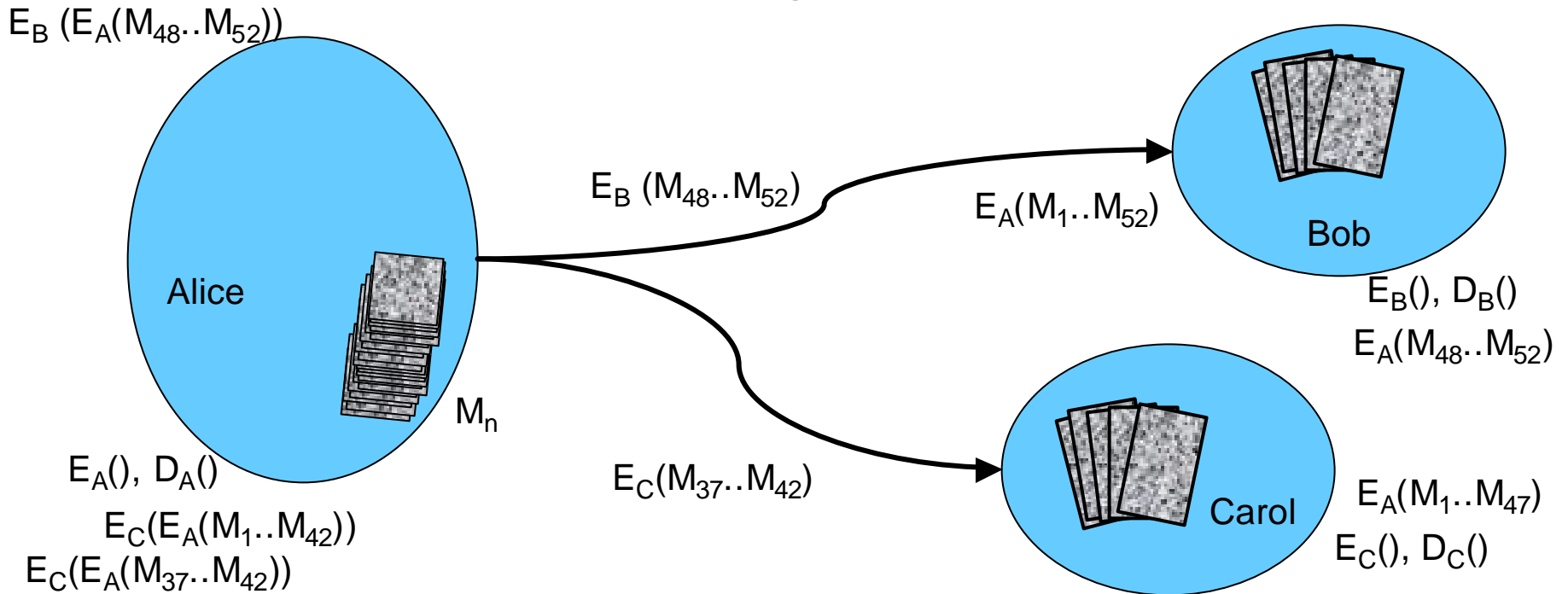
- Bob randomly picks 5 of the random messages and passes the other 47 to Carol. For ease of representation, say he picked the last 5
- Bob encrypts his 5 cards and sends them back to Alice

Implementing Mental Poker



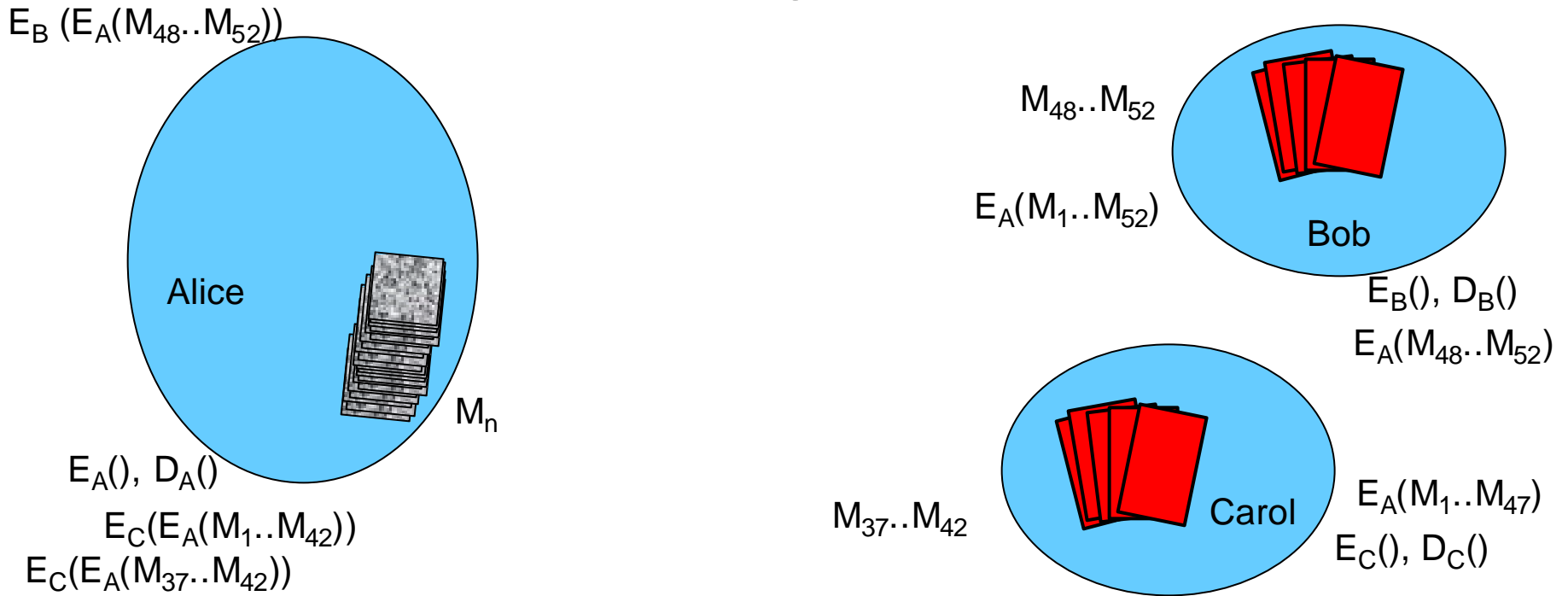
- Carol randomly picks 5 of the remaining random messages and passes the other 42 back to Alice after encrypting them. For ease of representation, say she also picked the last 5

Implementing Mental Poker



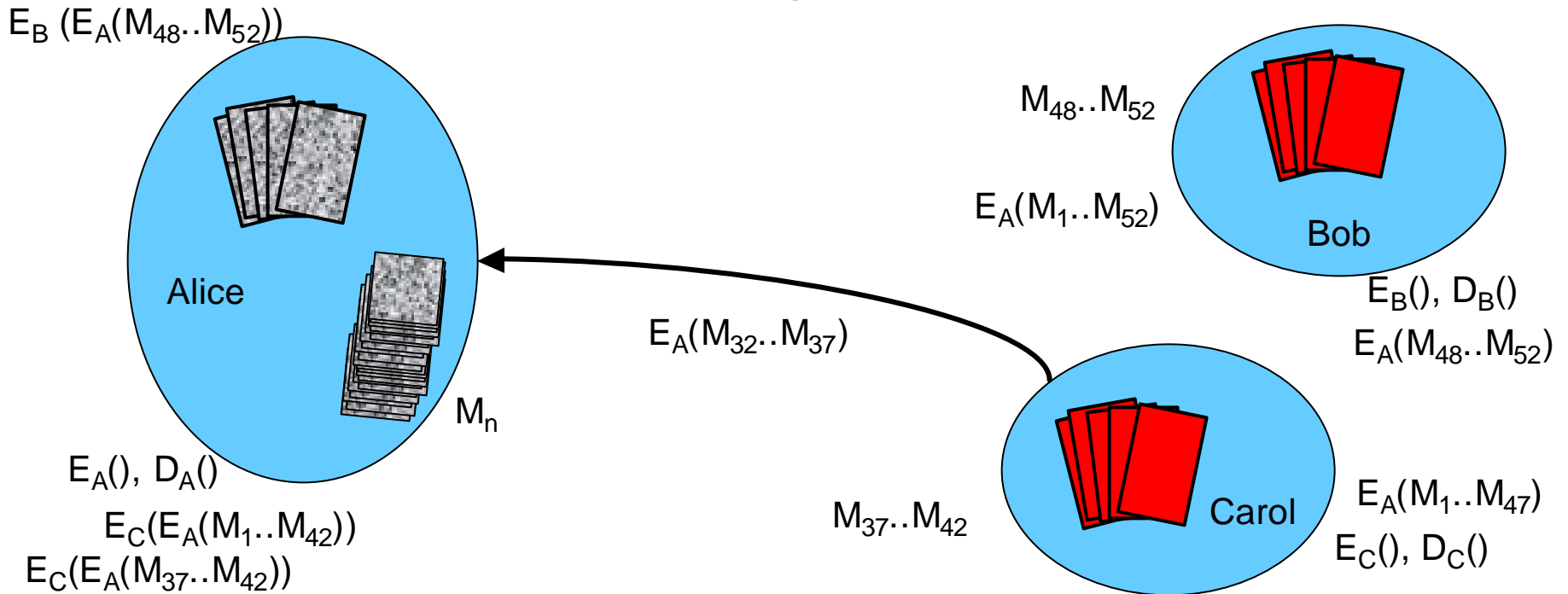
- Alice decrypts the messages with her key and sends the results back to Bob and Carol, respectively

Implementing Mental Poker



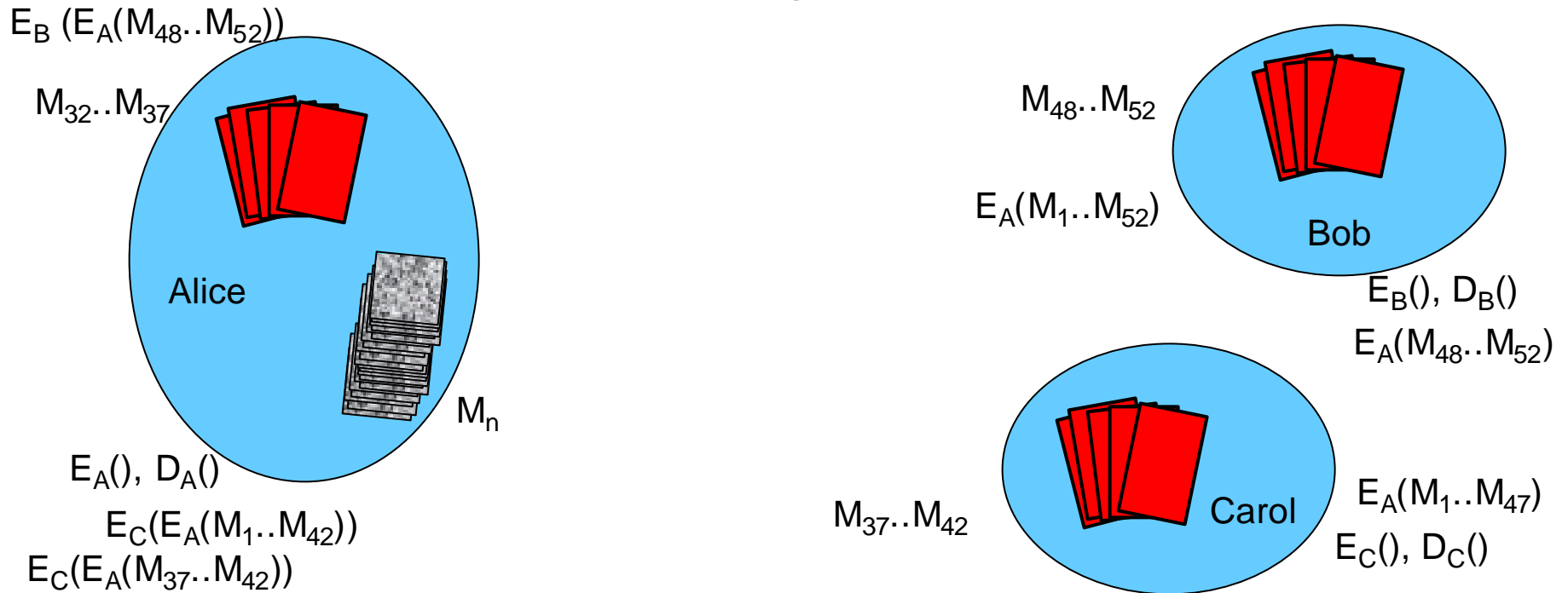
- Bob and Carol decrypt their messages to discover their hands

Implementing Mental Poker



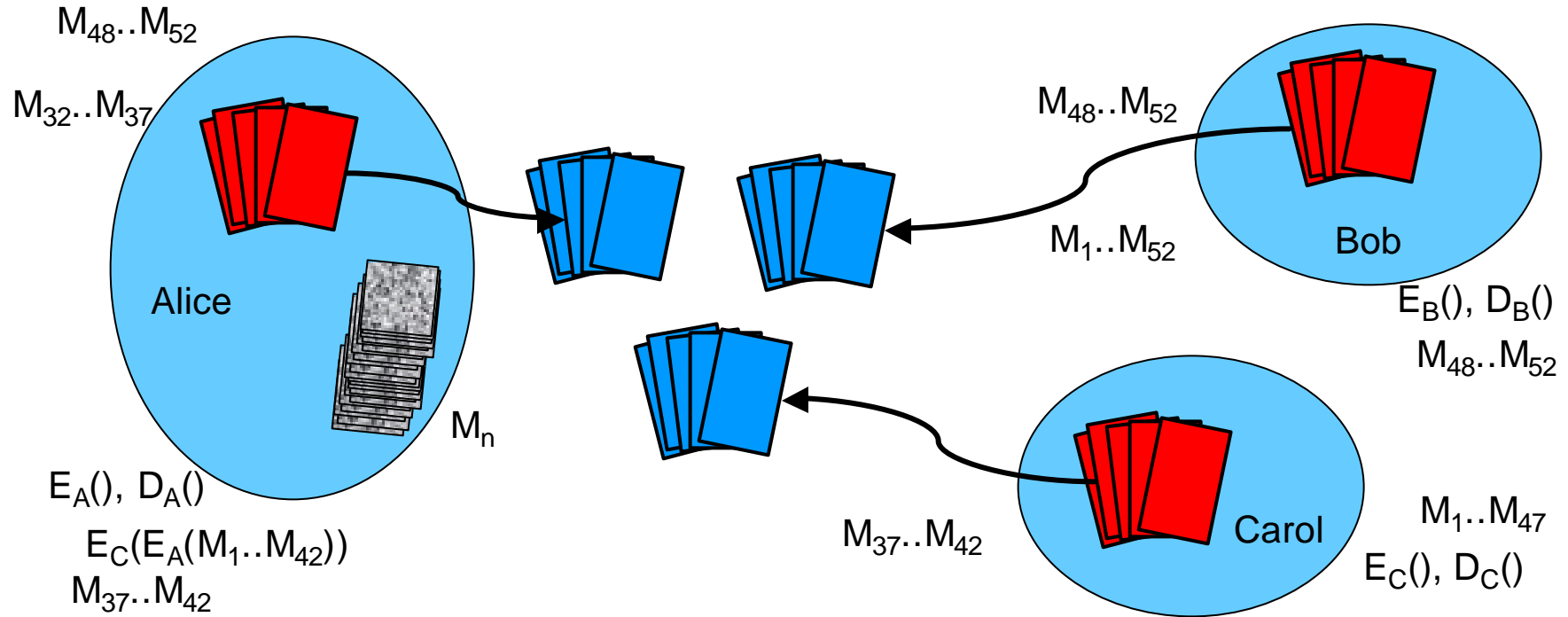
- Carol chooses 5 more cards randomly from the remaining 42 and sends them to Alice

Implementing Mental Poker



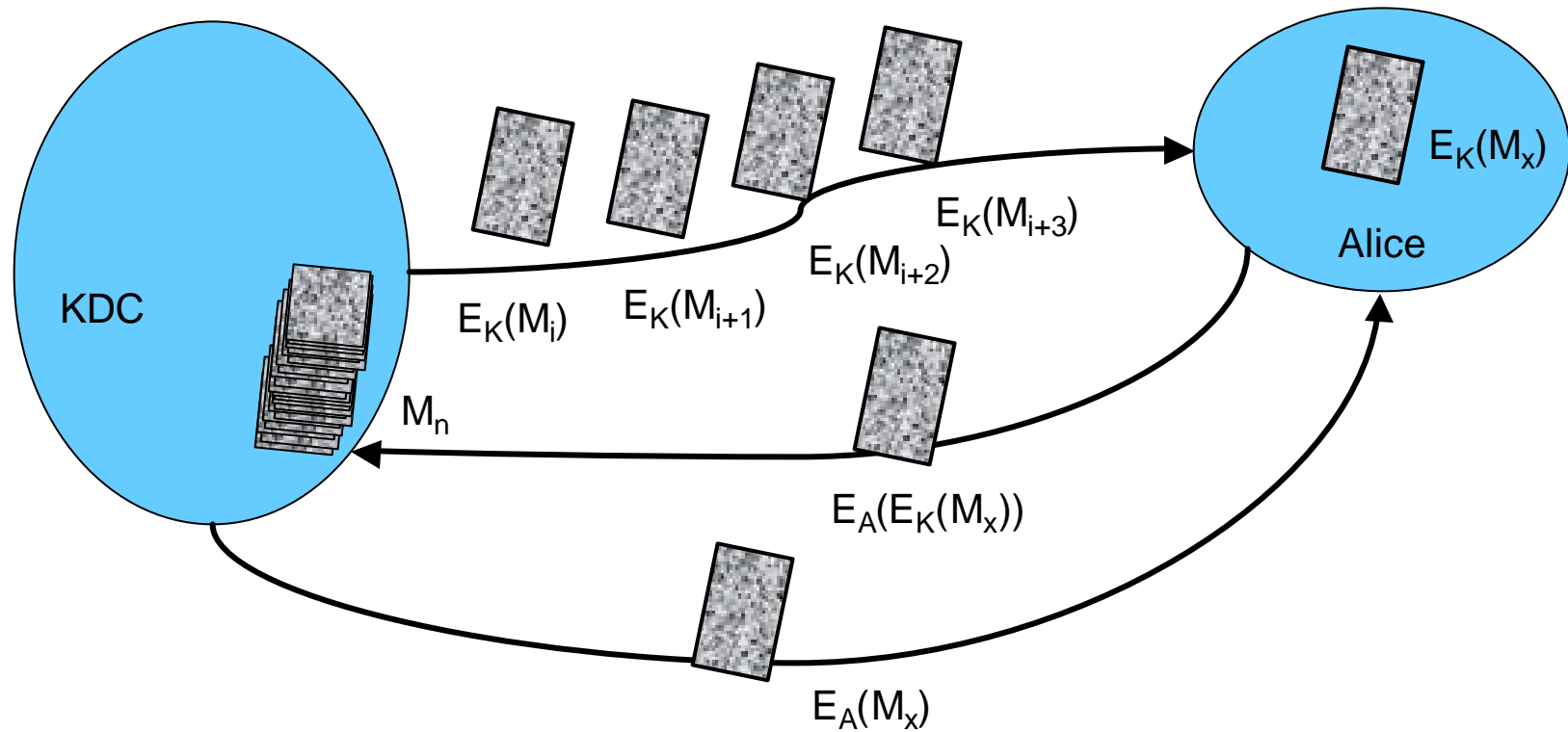
- Alice decrypts the message to discover her hand

Implementing Mental Poker



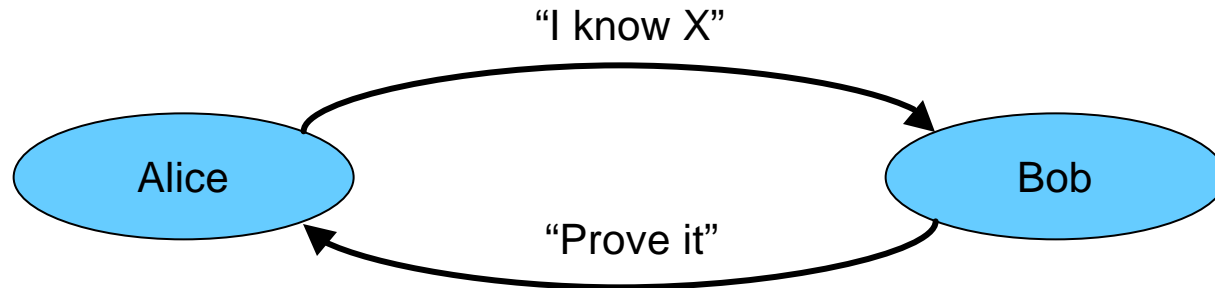
- Alice, Bob, and Carol reveal their keys, revealing their hands and that no one has cheated

Why Mental Poker?



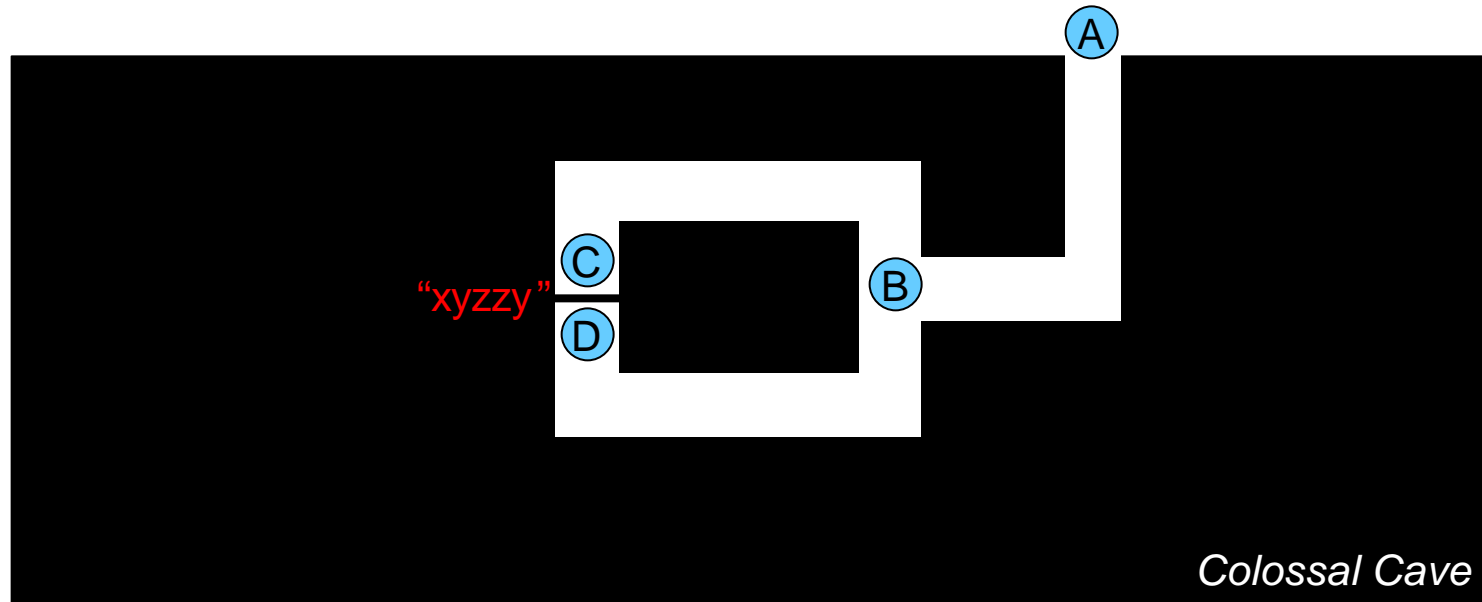
- Key distribution when the KDC cannot know the keys used.

Zero-Knowledge Proofs



- Alice needs to convince Bob that she has some information, X
- Bob wants proof that Alice has the information
- Conditions for *minimum-disclosure* proof:
 - Alice cannot cheat Bob:
 - She cannot convince Bob that she knows X unless she really does
 - Bob cannot cheat Alice:
 - He cannot learn the slightest information about the proof other than that Alice knows it
 - He cannot demonstrate the proof to anyone else without proving it himself
- Additional condition for *zero-knowledge* proof:
 - Bob learns nothing from Alice that he couldn't have learned without Alice

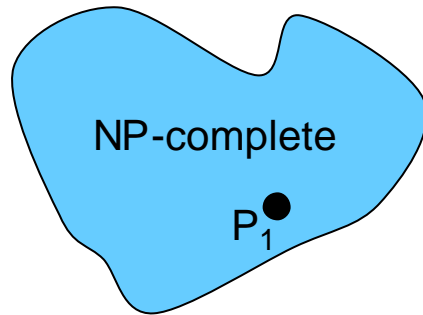
Zero-Knowledge Proofs - Physical Example



- Alice knows the secret password to open the gate between C and D and wants to prove her knowledge to Bob
- Bob waits at (A) while Alice enters cave and goes to *either* (C) or (D), randomly
- Bob goes to (B) and tells Alice to come out of either the right or left passage, randomly
- Alice opens the gate, if necessary, and comes out of the proper passage

- Alice's chances of guessing correctly, if she does not know the password is .5
- After repeating the protocol n times, her chances of always guessing correctly is 2^{-n}

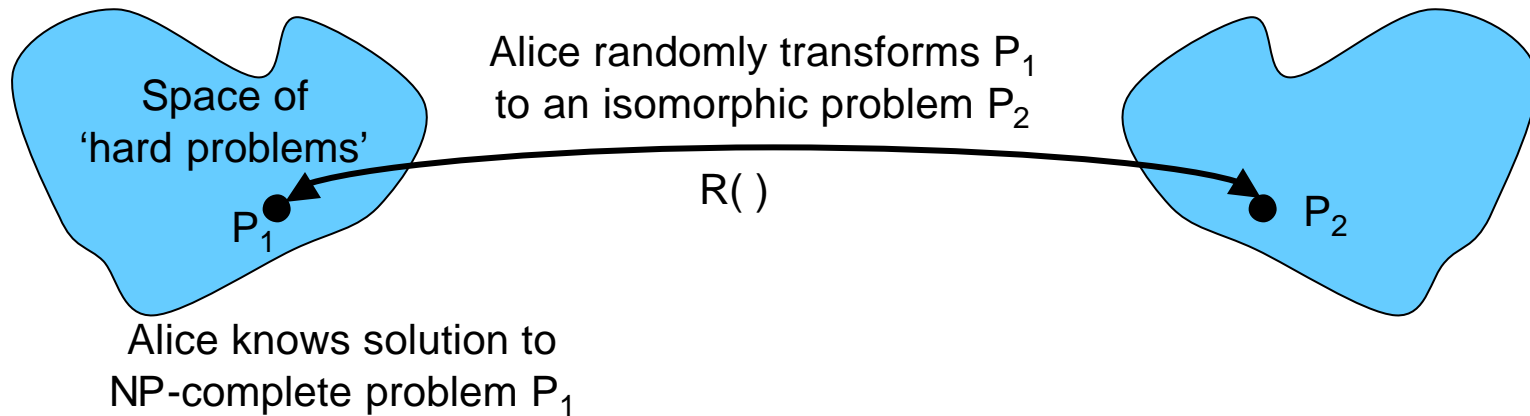
Implementing Zero-Knowledge Proofs



Alice knows solution to
NP-complete problem P_1

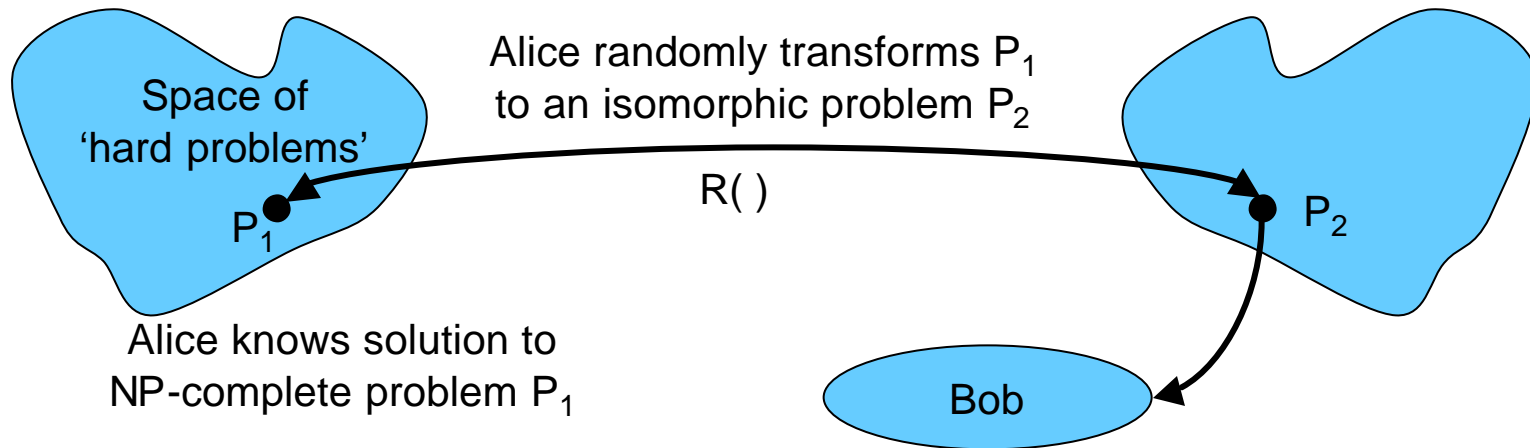
- Alice wants to prove she knows solution to P_1

Implementing Zero-Knowledge Proofs



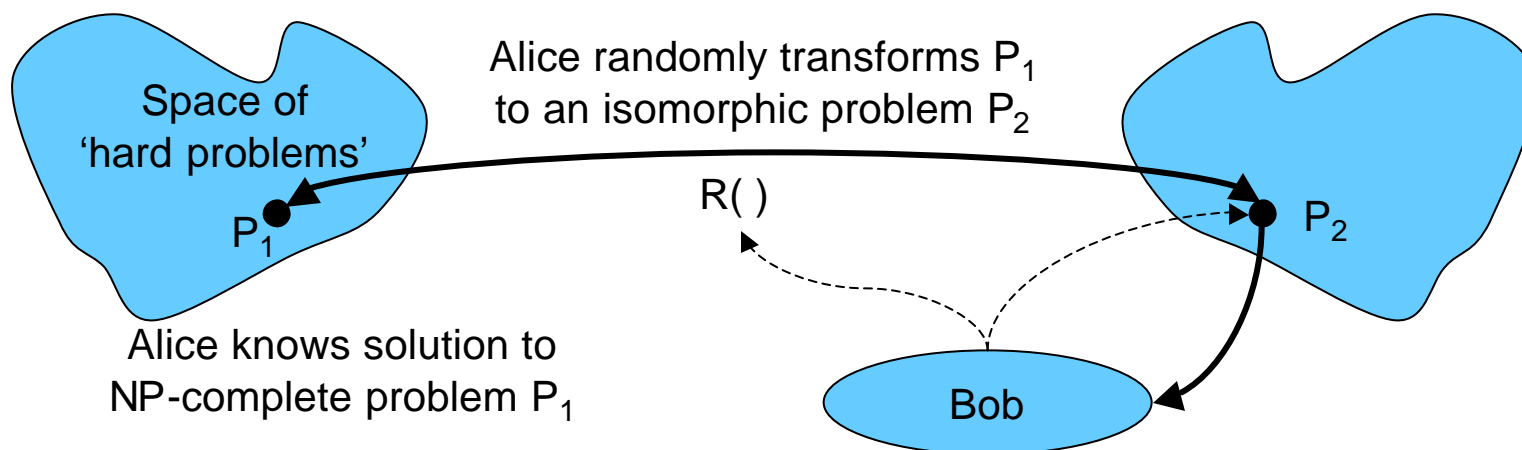
- Alice wants to prove she knows solution to P_1
- She transforms the problem to an equivalent problem which allows her to use the knowledge of P_1 and $R()$ to solve easily

Implementing Zero-Knowledge Proofs



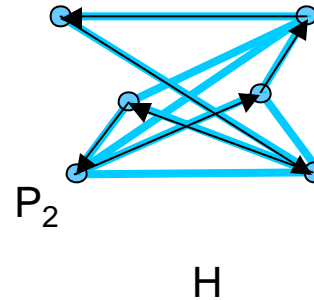
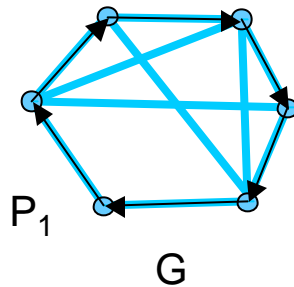
- Alice wants to prove she knows solution to P_1
- She transforms the problem to an equivalent problem which allows her to use the knowledge of P_1 and $R()$ to solve easily
- Alice uses a bit commitment scheme to commit to the solution of P_2

Implementing Zero-Knowledge Proofs



- Alice wants to prove she knows solution to P_1
- She transforms the problem to an equivalent problem which allows her to use the knowledge of P_1 and $R()$ to solve easily
- Alice uses a bit commitment scheme to commit to the solution of P_2
- Bob asks Alice to either
 - Prove that P_1 and P_2 are isomorphic or
 - Open the proof of P_2 to prove that Alice has proven P_2
- Protocol is repeated n times

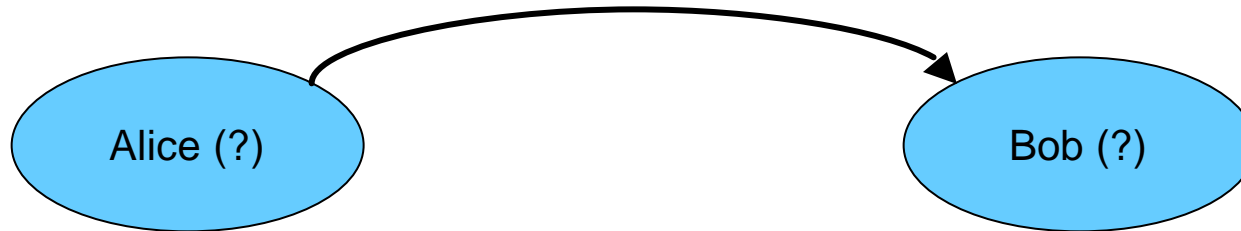
Implementing Zero-Knowledge Proofs



- P_1 and P_2 are Hamiltonian cycles in a graph, G
- Finding a Hamiltonian cycle on a graph is an NP-complete problem
- $H=R(G)$ is a random permutation of G
- Alice is asked either:
 - to demonstrate that she knows P_2 , a Hamiltonian cycle on H or
 - to demonstrate that G and H are isomorphic

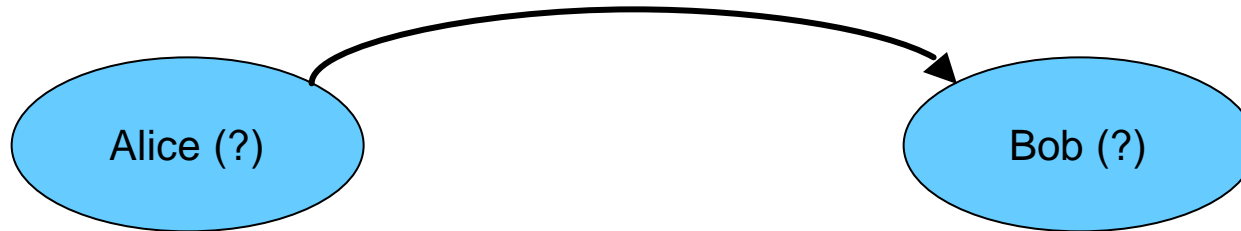
Zero-Knowledge Proofs of Identity

“I know Alice’s secret key.
Therefore, I am (indistinguishable from) Alice”



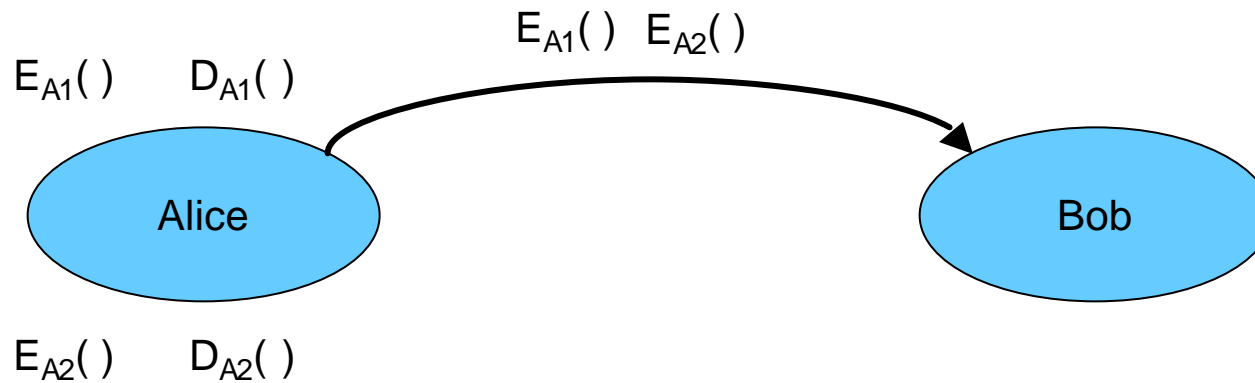
Zero-Knowledge Proofs of Identity

“I know Alice’s secret key.
Therefore, I am (indistinguishable from) Alice”



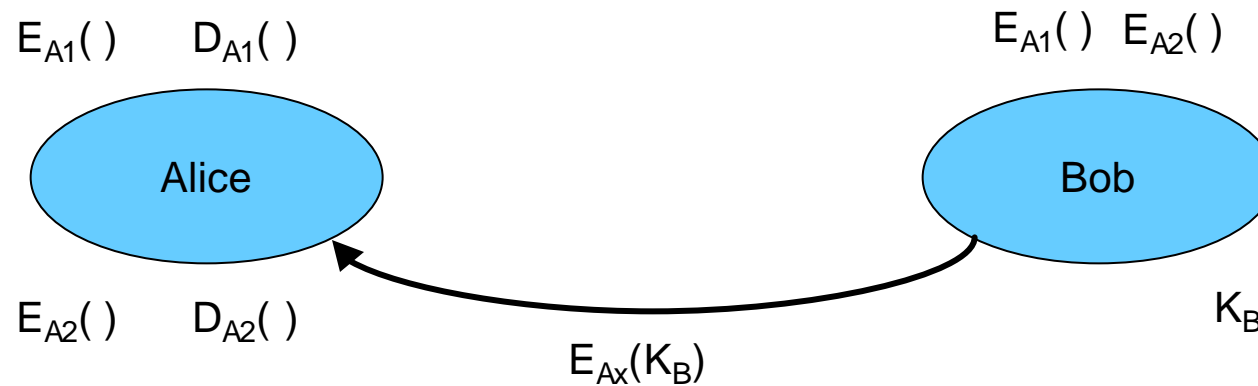
- How do you know that Alice has only one identity?

Oblivious Transfer



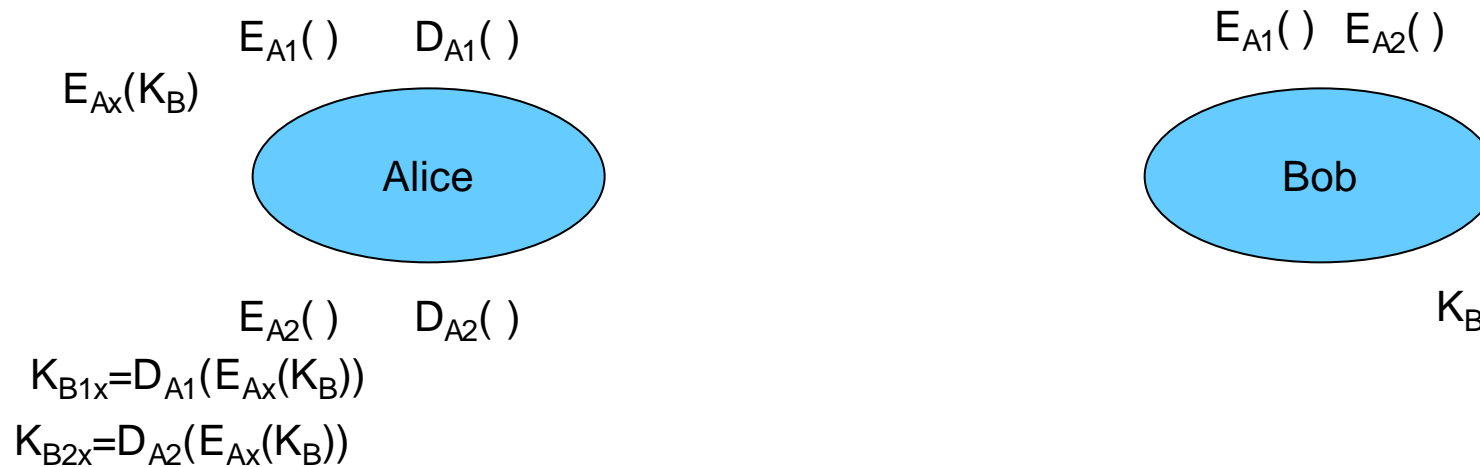
- Alice generates two PKC key pairs and sends the public half of each to Bob

Oblivious Transfer



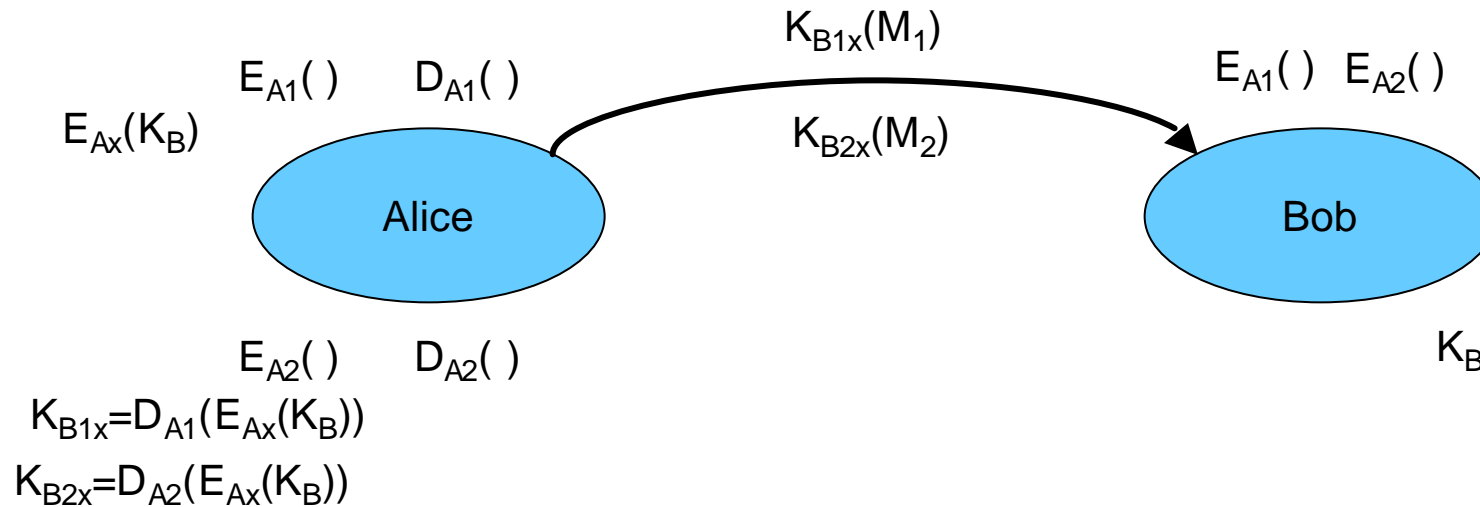
- Alice generates two PKC key pairs and sends the public half of each to Bob
- Bob generates a key for a symmetric cryptosystem, secretly picks one of Alices public keys and sends the encrypted key to Alice

Oblivious Transfer



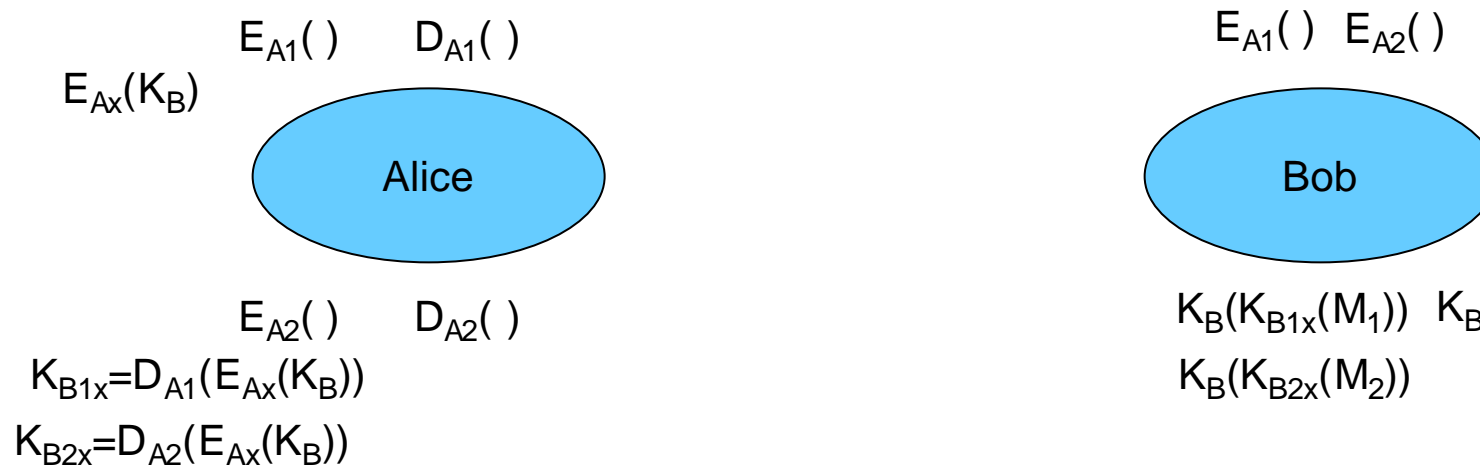
- Alice generates two PKC key pairs and sends the public half of each to Bob
- Bob generates a key for a symmetric cryptosystem, secretly picks one of Alices public keys and sends the encrypted key to Alice
- Not knowing which of her private keys to use, Alice decrypts Bob's message with both - providing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

Oblivious Transfer



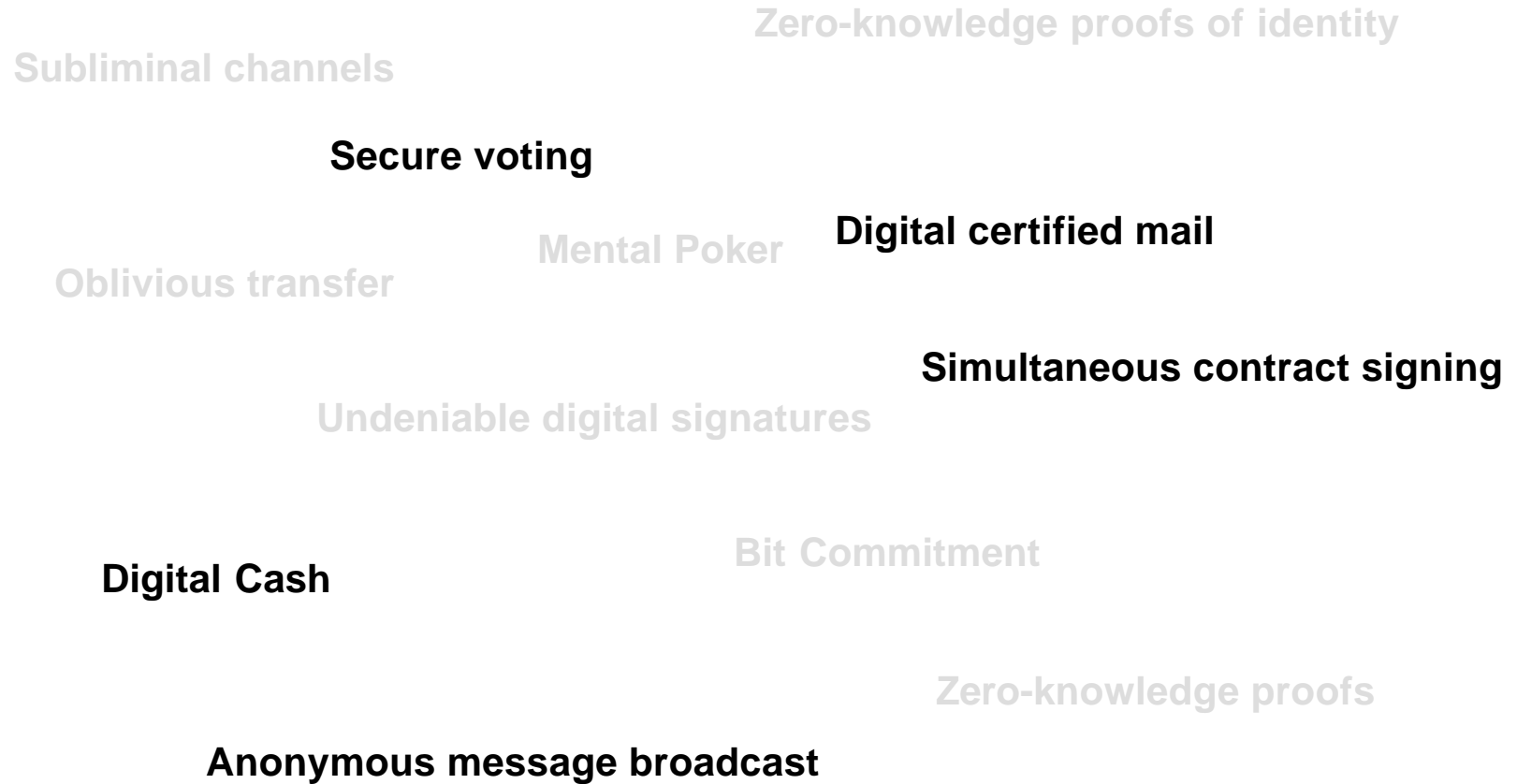
- Alice generates two PKC key pairs and sends the public half of each to Bob
- Bob generates a key for a symmetric cryptosystem, secretly picks one of Alices public keys and sends the encrypted key to Alice
- Not knowing which of her private keys to use, Alice decrypts Bob's message with both - providing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



- 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

Security Protocols - Next Time



Homework 10

- Proof of the knowledge of a Hamiltonian cycle on a graph and the isomorphic transformation of graphs has been shown as one method of performing a zero knowledge proof. Research and describe another.