1. Coin-Flipping

2. Locked Box Paradigm
   - Overview
   - Application to Coin-Flipping
   - Implementation

3. Oblivious Transfer
   - Oblivious Transfer of One Secret
   - Oblivious Transfer of One Secret Out of Two
Coin-Flipping
Flipping a common coin

Alice and Bob are in the process of getting divorced and are trying to decide who gets custody of their pet cat, Fluffy.
Alice and Bob are in the process of getting divorced and are trying to decide who gets custody of their pet cat, Fluffy.

They both want the cat, so they agree to decide by flipping a coin: heads Alice wins; tails Bob wins.
Alice and Bob are in the process of getting divorced and are trying to decide who gets custody of their pet cat, Fluffy.

They both want the cat, so they agree to decide by flipping a coin: heads Alice wins; tails Bob wins.

Bob has already moved out and does not wish to be in the same room with Alice.
Flipping a common coin

Alice and Bob are in the process of getting divorced and are trying to decide who gets custody of their pet cat, Fluffy.

They both want the cat, so they agree to decide by flipping a coin: heads Alice wins; tails Bob wins.

Bob has already moved out and does not wish to be in the same room with Alice.

The feeling is mutual, so Alice proposes that she flip the coin and telephone Bob with the result.
Alice and Bob are in the process of getting divorced and are trying to decide who gets custody of their pet cat, Fluffy.

They both want the cat, so they agree to decide by flipping a coin: heads Alice wins; tails Bob wins.

Bob has already moved out and does not wish to be in the same room with Alice.

The feeling is mutual, so Alice proposes that she flip the coin and telephone Bob with the result.

This proposal of course is not acceptable to Bob since he has no way of knowing whether Alice is telling the truth when she says that the coin landed heads.
“Look Alice,” he says, “to be fair, we both have to be involved in flipping the coin.”
“Look Alice,” he says, “to be fair, we both have to be involved in flipping the coin.”

“We’ll each flip a private coin and XOR our two coins together to determine who gets Fluffy.”
“Look Alice,” he says, “to be fair, we both have to be involved in flipping the coin.”

“We’ll each flip a private coin and XOR our two coins together to determine who gets Fluffy.”

“You should be happy with this arrangement since even if you don’t trust me to flip fairly, your own fair coin is sufficient to ensure that the XOR is unbiased.”
### A proposed protocol

This sounds reasonable to Alice, so she lets him propose the protocol below, where 1 means “heads” and 0 means “tails”.

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Choose random bit ( b_A \in {0, 1} )</td>
<td>Choose random bit ( b_B \in {0, 1} ).</td>
</tr>
<tr>
<td>( b = b_A \oplus b_B ).</td>
<td>( b = b_A \oplus b_B ).</td>
</tr>
</tbody>
</table>

Alice considers this for awhile, then objects. “This isn’t fair. You get to see my coin before I see yours, so now you have complete control over the outcome.”
A proposed protocol

This sounds reasonable to Alice, so she lets him propose the protocol below, where 1 means “heads” and 0 means “tails”.

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Choose random bit $b_A \in {0, 1}$</td>
<td>$b_A \rightarrow$</td>
</tr>
<tr>
<td>2. $b_B \leftarrow$ Choose random bit $b_B \in {0, 1}$.</td>
<td></td>
</tr>
<tr>
<td>3. Coin outcome is $b = b_A \oplus b_B$.</td>
<td>Coin outcome is $b = b_A \oplus b_B$.</td>
</tr>
</tbody>
</table>

Alice considers this for awhile, then objects.

“This isn’t fair. You get to see my coin before I see yours, so now you have complete control over the outcome.”
Alice’s counter proposal

She suggests that she would be happy if the first two steps were reversed, so that Bob flips his coin first, but Bob balks at that suggestion.
Alice’s counter proposal

She suggests that she would be happy if the first two steps were reversed, so that Bob flips his coin first, but Bob balks at that suggestion.

They then both remember Lecture 23 and decide to use blobs to prevent either party from controlling the outcome. They agree on the following protocol.
A mutually acceptable protocol

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Choose random $k_A, s_A \in K_A$.</td>
<td>Choose random $k_B, s_B \in K_B$.</td>
</tr>
<tr>
<td>2. Choose random bit $b_A \in {0, 1}$.</td>
<td>Choose random bit $b_B \in {0, 1}$.</td>
</tr>
<tr>
<td>$c_A = \text{enclose}(s_A, k_B, b_A)$.</td>
<td>$c_B = \text{enclose}(s_B, k_A, b_B)$.</td>
</tr>
<tr>
<td>3. Send $s_A$.</td>
<td>Send $s_B$.</td>
</tr>
<tr>
<td>4. $b_B = \text{reveal}(s_B, k_A, c_B)$.</td>
<td>$b_A = \text{reveal}(s_A, k_B, c_A)$.</td>
</tr>
<tr>
<td>Coin outcome is $b = b_A \oplus b_B$.</td>
<td>Coin outcome is $b = b_A \oplus b_B$.</td>
</tr>
</tbody>
</table>
# A mutually acceptable protocol

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Choose random $k_A, s_A \in \mathcal{K}_A$.</td>
<td>Choose random $k_B, s_B \in \mathcal{K}_B$.</td>
</tr>
<tr>
<td>2. Choose random bit $b_A \in {0, 1}$. $c_A = \text{enclose}(s_A, k_B, b_A)$.</td>
<td>Choose random bit $b_B \in {0, 1}$. $c_B = \text{enclose}(s_B, k_A, b_B)$.</td>
</tr>
<tr>
<td>3. Send $s_A$.</td>
<td>Send $s_B$.</td>
</tr>
<tr>
<td>4. $b_B = \text{reveal}(s_B, k_A, c_B)$. Coin outcome is $b = b_A \oplus b_B$.</td>
<td>$b_A = \text{reveal}(s_A, k_B, c_A)$. Coin outcome is $b = b_A \oplus b_B$.</td>
</tr>
</tbody>
</table>

At the completion of step 2, both Alice and Bob have each others commitment
A mutually acceptable protocol

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Choose random $k_A, s_A \in \mathcal{K}_A$.</td>
<td>$k_A, k_b \leftarrow \rightarrow$ Choose random $k_B, s_B \in \mathcal{K}_B$.</td>
</tr>
<tr>
<td>2. Choose random bit $b_A \in {0, 1}$. $c_A = \text{enclose}(s_A, k_B, b_A)$.</td>
<td>$c_B = \text{enclose}(s_B, k_A, b_B)$.</td>
</tr>
<tr>
<td>3. Send $s_A$.</td>
<td>$s_A, s_B \leftarrow \rightarrow$ Send $s_B$.</td>
</tr>
<tr>
<td>4. $b_B = \text{reveal}(s_B, k_A, c_B)$. Coin outcome is $b = b_A \oplus b_B$.</td>
<td>$b_A = \text{reveal}(s_A, k_B, c_A)$. Coin outcome is $b = b_A \oplus b_B$.</td>
</tr>
</tbody>
</table>

At the completion of step 2, both Alice and Bob have each others commitment (something they failed to achieve in the past, which is why they’re in the middle of a divorce now)
A mutually acceptable protocol

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Choose random $k_A, s_A \in \mathcal{K}_A.$</td>
<td>Choose random $k_B, s_B \in \mathcal{K}_B.$</td>
</tr>
<tr>
<td>2. Choose random bit $b_A \in {0, 1}.$</td>
<td>Choose random bit $b_B \in {0, 1}.$</td>
</tr>
<tr>
<td>$c_A = \text{enclose}(s_A, k_B, b_A).$</td>
<td>$c_B = \text{enclose}(s_B, k_A, b_B).$</td>
</tr>
<tr>
<td>3. Send $s_A.$</td>
<td>Send $s_B.$</td>
</tr>
<tr>
<td>4. $b_B = \text{reveal}(s_B, k_A, c_B).$</td>
<td>$b_A = \text{reveal}(s_A, k_B, c_A).$</td>
</tr>
<tr>
<td>Coin outcome is $b = b_A \oplus b_B.$</td>
<td>Coin outcome is $b = b_A \oplus b_B.$</td>
</tr>
</tbody>
</table>

At the completion of step 2, both Alice and Bob have each others commitment (something they failed to achieve in the past, which is why they’re in the middle of a divorce now), but neither knows the other’s private bit.

They learn each other’s bit at the completion of steps 3 and 4.
While this protocol appears to be completely symmetric, it really isn’t quite, for one of the parties completes step 3 before the other one does.

Say Alice receives $s_B$ before sending $s_A$.

At that point, she can compute $b_B$ and hence know the coin outcome $b$.

If it turns out that she lost, she might decide to stop the protocol and refuse to complete her part of step 3.
Premature termination

What happens if one party quits in the middle or detects the other party cheating?

So far, we’ve only considered the possibility of undetected cheating. But in any real situation, one party might feel that he or she stands to gain by cheating, *even if the cheating is detected.*
Responses to cheating

Detected cheating raises complicated questions as to what happens next.

- Does a third party Carol become involved?
- If so, can Bob prove to Carol that Alice cheated?
- What if Alice refuses to talk to Carol?

Think about Bob’s recourse in similar real-life situations and consider the reasons why such situations rarely arise.

For example, what happens if someone

- fails to follow the provisions of a contract?
- ignores a summons to appear in court?
A copycat attack

There is a subtle problem with the previous coin-flipping protocol. Suppose Bob sends his message before Alice sends hers in each of steps 1, 2, and 3.

Then Alice can choose $k_A = k_B$, $c_A = c_B$, and $s_A = s_B$ rather than following her proper protocol, so

$$\text{reveal}(s_A, k_B, c_A) = \text{reveal}(s_B, k_A, c_B).$$

In step 4, Bob will compute $b_A = b_b$ and won’t detect that anything is wrong. The coin outcome is $b = b_A \oplus b_A = 0$.

Hence, Alice can force outcome 0 simply by playing copycat.
Preventing a copycat attack

This problem is not so easy to overcome.

One possibility is for both Alice and Bob to check that $k_A \neq k_B$ after step 1.

That way, if Alice, say, chooses $c_A = c_B = c$ and $s_A = s_B = s$ on steps 2 and 3, there still might be a good chance that

$$b_A = \text{reveal}(s, k_B, c) \neq \text{reveal}(s, k_A, c) = b_B.$$ 

However, depending on the bit commitment scheme, a difference in only one bit in $k_A$ and $k_B$ might not be enough to ensure that different bits are revealed.

In any case, it’s not enough that $b_A$ and $b_B$ sometimes differ. For the outcome to be unbiased, we need $P[b_A \neq b_B] = 1/2.$
A better idea might be to both check that $k_A \neq k_B$ after step 1 and then to use $h(k_A)$ and $h(k_B)$ in place of $k_A$ and $k_B$, respectively, in the remainder of the protocol, where $h$ is a hash function.

That way, even a single bit difference in $k_A$ and $k_B$ is likely to be magnified to a large difference in the strings $h(k_A)$ and $h(k_B)$.

This should lead to the bits $\text{reveal}(s_A, h(k_B), c_A)$ and $\text{reveal}(s_B, h(k_A), c_B)$ being uncorrelated, even if $s_A = s_B$ and $c_A = c_B$. 

Locked Box Paradigm
Protocols for coin flipping and for dealing a poker hand from a deck of cards can be based on the intuitive notion of locked boxes. This idea in turn can be implemented using commutative-key cryptosystems.

We first present a coin-flipping protocol using locked boxes.
Preparing the boxes

Imagine two sturdy boxes with hinged lids that can be locked with a padlock.

Alice writes “heads” on a slip of paper and “tails” on another.

“heads”, signed Alice  
“tails”, signed Alice

She places one of these slips in each box.
Alice locks the boxes

Alice puts a padlock on each box for which she holds the only key.

She then gives both locked boxes to Bob, in some random order.
Bob adds his lock

Bob cannot open the boxes and does not know which box contains “heads” and which contains “tails”.

He chooses one of the boxes and locks it with his own padlock, for which he has the only key.

He gives the doubly-locked box back to Alice.
Alice removes her lock

Alice gets

She removes her lock.

and returns the box to Bob.
Bob opens the box

Bob gets

He removes his lock

opens the box, and removes the slip of paper from inside.

“heads”, signed Alice

He gives the slip to Alice.
Alice checks that Bob didn’t cheat

At this point, both Alice and Bob know the outcome of the coin toss.

Alice verifies that the slip of paper is one of the two that she prepared at the beginning, with her handwriting on it.

She sends her key to Bob.
Bob check that Alice didn’t cheat

Bob still has the other box.

He removes Alice’s lock,

opens the box, and removes the slip of paper from inside.

“tails”, signed Alice

He checks that it contains the other coin value.
Implementation
Commutative-key cryptosystems

Alice and Bob can carry out this protocol electronically using any commutative-key cryptosystem, that is, one in which $E_A \circ E_B = E_B \circ E_A$.\(^1\)

RSA is commutative for keys $A$ and $B$ with a common modulus $n$, so we can use RSA in an unconventional way.

Rather than making the encryption exponent public and keeping the factorization of $n$ private, we turn things around.

\(^1\)Recall the related notion of “commutative cryptosystem” of Lecture 14 in which the encryption and decryption functions for the same key commuted.
RSA as a commutative-key cryptosystem

Alice and Bob jointly chose primes $p$ and $q$, and both compute $n = pq$.

Alice chooses an RSA key pair $A = ((e_A, n), (d_A, n))$, which she can do since she knows the factorization of $n$.

Similarly, Bob chooses an RSA key pair $B = ((e_B, n), (d_B, n))$ using the same $n$.

Alice and Bob both keep their key pairs private (until the end of the protocol, when they reveal them to each other to verify that there was no cheating).
Security remark

We note that this scheme may have completely different security properties from usual RSA.

In RSA, there are *three different secrets* involved with the key: the factorization of $n$, the encryption exponent $e$, and the decryption exponent $d$.

We have seen previously that knowing $n$ and any two of these three pieces of information allows the third to be reconstructed.

Thus, knowing the factorization of $n$ and $e$ lets one compute $d$. We also showed in Lecture 11 how to factor $n$ given both $e$ and $d$.

The way RSA is usually used, only $e$ is public, and it is believed to be hard to find the other two secrets.
Here we propose making the factorization of $n$ public but keeping $e$ and $d$ private.

It may indeed be hard to find $e$ and $d$, even knowing the factorization of $n$, but if it is, that fact is not going to follow from the difficulty of factoring $n$.

Of course, for security, we need more than just that it is hard to find $e$ and $d$.

We also need it to be hard to find $m$ given $c = m^e \mod n$.

This is reminiscent of the discrete log problem, but of course $n$ is not prime in this case.
We now implement the locked box protocol using RSA.

Here we assume that Alice and Bob initially know large primes $p$ and $q$.

In step (2), Alice chooses a random number $r$ such that $r < (n − 1)/2$.

This ensures that $m_0$ and $m_1$ are both in $\mathbb{Z}_n$.

Note that $i$ and $r$ can be efficiently recovered from $m_i$ since $i$ is just the low-order bit of $m_i$ and $r = (m_i − i)/2$. 
<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Choose RSA key pair $A$ with modulus $n = pq$.</td>
<td>Choose RSA key pair $B$ with modulus $n = pq$.</td>
</tr>
<tr>
<td>2. Choose random $r \in \mathbb{Z}_{(n-1)/2}$. Let $m_i = 2r + i$, for $i \in {0, 1}$. Let $c_i = E_A(m_i)$ for $i \in {0, 1}$. Let $C = {c_0, c_1}$.</td>
<td>$\rightarrow C$ Choose $c_a \in C$.</td>
</tr>
<tr>
<td>3. $\leftarrow c_{ab}$ Let $c_{ab} = E_B(c_a)$.</td>
<td></td>
</tr>
<tr>
<td>4. Let $c_b = D_A(c_{ab})$.</td>
<td>$\rightarrow c_b$</td>
</tr>
<tr>
<td>5. Let $m = D_B(c_b)$. Let $i = m \mod 2$. Let $r = (m - i)/2$. If $i = 0$ then “tails”. If $i = 1$ then “heads”.</td>
<td>$\leftarrow B$</td>
</tr>
<tr>
<td>Alice</td>
<td>Bob</td>
</tr>
<tr>
<td>-------</td>
<td>-----</td>
</tr>
</tbody>
</table>
| 6. Let $m = D_B(c_b)$.  
Check $m \in \{m_0, m_1\}$.  
If $m = m_0$ then “tails”.  
If $m = m_1$ then “heads”. | 
\[
\longrightarrow A
\]
| 7. Let $c'_a = C - \{c_a\}$.  
Let $m' = D_A(c'_a)$.  
Let $i' = m' \mod 2$.  
Let $r' = (m' - i')/2$.  
Check $i' \neq i$ and $r' = r$. |
Correctness when Alice and Bob are honest

When both Alice and Bob are honest, Bob computes $c_{ab} = E_B(E_A(m_j))$ for some $j \in \{0, 1\}$.

In step 4, Alice computes $c_b$.

By the commutativity of $E_A$ and $E_B$,

$$c_b = D_A(E_B(E_A(m_j))) = E_B(m_j).$$

Hence, in step 5, $m = m_j$ is one of Alice’s strings from step 2.
A dishonest Bob

A dishonest Bob can control the outcome of the coin toss if he can find two keys $B$ and $B'$ such that $E_B(c_a) = E_{B'}(c'_a)$, where $C = \{c_a, c'_a\}$ is the set received from Alice in step 2.

In this case, $c_{ab} = E_B(E_A(m_j)) = E_{B'}(E_A(m_{1-j}))$ for some $j$. Then in step 4, $c_b = D_A(c_{ab}) = E_B(m_j) = E_{B'}(m_{1-j})$.

Hence, $m_j = D_B(c_b)$ and $m_{1-j} = D_{B'}(c_b)$, so Bob can obtain both of Alice’s messages and then send $B$ or $B'$ in step 5 to force the outcome to be as he pleases.

To find such $B$ and $B'$, Bob would need to solve the equation

$$c_a^e \equiv c'_a^{e'} \pmod{n}$$

for $e$ and $e'$. Not clear how to do this, even knowing the factorization of $n$. 
Card dealing using locked boxes

The same locked box paradigm can be used for dealing a 5-card poker hand from a deck of cards.

Alice takes a deck of cards, places each card in a separate box, and locks each box with her lock.

She arranges the boxes in random order and ships them off to Bob.

Bob picks five boxes, locks each with his lock, and send them back.

Alice removes her locks from those five boxes and returns them to Bob.

Bob unlocks them and obtains the five cards of his poker hand.

Further details are left to the reader.
Oblivious Transfer
In the locked box coin-flipping protocol, Alice has two messages $m_0$ and $m_1$.

Bob gets one of them.

Alice doesn’t know which (until Bob tells her).

Bob can’t cheat to get both messages.

Alice can’t cheat to learn which message Bob got.

The *oblivious transfer problem* abstracts these properties from particular applications such as coin flipping and card dealing,
Oblivious Transfer of One Secret
Alice has a secret $s$.

An oblivious transfer protocol has two equally-likely outcomes:

1. Bob learns $s$.
2. Bob learns nothing.

Afterwards, Alice doesn’t know whether or not Bob learned $s$.

A cheating Bob can do nothing to increase his chance of getting $s$.

A cheating Alice can do nothing to learn whether or not Bob got her secret.

Rabin proposed an oblivious transfer protocol based on quadratic residuosity in the early 1980’s.
Rabin’s OT protocol

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1.</strong> Secret $s$.</td>
<td></td>
</tr>
<tr>
<td>$n = pq$, $p \neq q$ prime.</td>
<td></td>
</tr>
<tr>
<td>RSA public key $(e, n)$.</td>
<td></td>
</tr>
<tr>
<td>Compute $c = E_{(e,n)}(s)$.</td>
<td>$(e,n,c) \rightarrow$</td>
</tr>
<tr>
<td><strong>2.</strong> Choose random $x \in \mathbb{Z}_n^*$.</td>
<td>Compute $a = x^2 \mod n$.</td>
</tr>
<tr>
<td><strong>3.</strong> Check $a \in \mathbb{QR}_n$.</td>
<td></td>
</tr>
<tr>
<td>Random $y \in \sqrt{a} \pmod{n}$.</td>
<td>$y \rightarrow$</td>
</tr>
<tr>
<td><strong>4.</strong> Check $y^2 \equiv a \pmod{n}$.</td>
<td></td>
</tr>
<tr>
<td>If $y \not\equiv \pm x \pmod{n}$, use $x, y$ to factor $n$ and decrypt $c$ to obtain $s$.</td>
<td></td>
</tr>
</tbody>
</table>
Alice can carry out step 3 since she knows the factorization of $n$ and can find all four square roots of $a$.

However, Alice has no idea which $x$ Bob used to generate $a$.

Hence, with probability $1/2$, $y \equiv \pm x \pmod{n}$ and with probability $1/2$, $y \not\equiv \pm x \pmod{n}$.

If $y \not\equiv \pm x \pmod{n}$, then the two factors of $n$ are $\gcd(x - y, n)$ and $n/\gcd(x - y, n)$, so Bob factors $n$ and decrypts $c$ in step 4.

However, if $y \equiv \pm x \pmod{n}$, Bob learns nothing, and Alice’s secret is as secure as RSA itself.
A potential problem

There is a potential problem with this protocol.

A cheating Bob in step 2 might send a number $a$ which he generated by some means other than squaring a random $x$.

In this case, he always learns something new no matter which square root Alice sends him in step 3.

Perhaps that information, together with what he already learned in the course of generating $a$, is enough for him to factor $n$. 
Is this a real problem?

We don’t know of any method by which Bob can find a quadratic residue $a \pmod{n}$ without also knowing one of $a$’s square roots.

We certainly don’t know of any method that would produce a quadratic residue $a$ together with some other information $\Xi$ that, combined with a square root $y$, would allow Bob to factor $n$.

But we also cannot prove that no such method exists.
A modified protocol

We fix this problem by having Bob prove that he knows a square root of the number $a$ that he sends Alice in step 2.

He does this using a zero knowledge proof of knowledge of a square root of $a$.

This is essentially what the simplified Feige-Fiat-Shamir protocol of Lecture 18 does, but with the roles of Alice and Bob reversed.

- Bob claims to know a square root $x$ of the public number $a$.
- He wants to prove to Alice that he knows $x$, but he does not want Alice to get any information about $x$.
- If Alice were to learn $x$, then she could choose $y = x$ and eliminate Bob’s chance of learning $s$ while still appearing to play honestly.
Oblivious Transfer of One Secret Out of Two
In *one-of-two oblivious transfer*, Alice has two secrets, $s_0$ and $s_1$. Bob always gets exactly one of the secrets, each with probability $1/2$.

Alice does not know which one Bob gets.

The locked box protocol is one way to implement one-of-two oblivious transfer.

Another is based on a public key cryptosystem (such as RSA) and a symmetric cryptosystem (such as AES).

This protocol given next does not rely on the cryptosystems being commutative.
# A one-of-two OT protocol

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1.</strong></td>
<td><strong>2.</strong></td>
</tr>
<tr>
<td>Secrets $s_0$ and $s_1$. Choose two PKS key pairs $(e_0, d_0)$ and $(e_1, d_1)$.</td>
<td>Choose key $k$ for symmetric cryptosystem $(\hat{E}, \hat{D})$. Choose random $b \in {0, 1}$. Compute $c = E_{eb}(k)$.</td>
</tr>
<tr>
<td><strong>3.</strong> Let $k_i = D_{d_i}(c)$, $i \in {0, 1}$. Choose $b' \in {0, 1}$. Let $c_i = \hat{E}<em>{k_i}(s</em>{i \oplus b'})$, $i \in {0, 1}$.</td>
<td></td>
</tr>
</tbody>
</table>
| **4.** Output $s = s_{b \oplus b'} = \hat{D}_k(c_b)$.
Analysis

In step 2, Bob encrypts a randomly chosen key $k$ for the symmetric cryptosystem using one of the PKS encryption keys that Alice sent him in step 1.

He then selects one of the two encryption keys from Alice, uses it to encrypt $k$, and sends the encryption to Alice.

In step 3, Alice decrypts $c$ using both decryption keys $d_0$ and $d_1$ to get $k_0$ and $k_1$.

One of the $k_i$ is Bob’s key $k$ ($k_b$ to be specific) and the other is garbage, but because $k$ is random and she doesn’t know $b$, she can’t tell which is $k$.

She then encrypts one secret with $k_0$ and the other with $k_1$, using the random bit $b'$ to ensure that each secret is equally likely to be encrypted by the key that Bob knows.
In step 4, Bob decrypts the ciphertext $c_b$ using key his key $k = k_b$ to recover the secret $s = s_b \oplus b'$.

He can’t decrypt the other ciphertext $c_1 \oplus b$ since he doesn’t know the key $k_1 \oplus b$ used to produce it, nor does he know the decryption key $d_1 \oplus b$ that would allow him to find it from $c$. 