CPSC 467: Cryptography and Security

Michael J. Fischer

Lecture 23
November 19, 2020

Oblivious Transfer

Locked-box protocol revisited

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

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

	Alice		Bob
1.	Secret s. $n = pq$, $p \neq q$ prime. RSA public key (e, n). Compute $c = E_{(e,n)}(s)$.	$\xrightarrow{(e,n,c)}$	
2.		$\xleftarrow{a}$	Choose random $x \in \mathrm{Z}_n^*$. Compute $a = x^2 \bmod n$.
3.	Check $a \in \mathrm{QR}_n$. Random $y \in \sqrt{a} \pmod n$.	$\xrightarrow{y}$	
4.			Check $y^2 \equiv a \pmod n$. If $y \not\equiv \pm x \pmod n$, use x, y to factor n and decrypt c to obtain s.

Analysis

Alice can 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 Ξ 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

One-of-two oblivious transfer

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.

Intuitive idea

In this protocol, Alice chooses two PKS key pairs and sends the public keys to Bob.

Bob chooses a random key k for a symmetric cryptosystem, encrypts it with one of Alice's two keys chosen at random, and sends the ciphertext to Alice.

Alice decrypts Bob's ciphertext using both of her decryption keys and obtains two numbers $\{k_0, k_1\}$. One of them is Bob's k; the other is garbage. She encrypts one secret using k_0 and one using k_1 and sends to Bob.

Bob decrypts the one that was encrypted with k.

A one-of-two OT protocol

	Alice		Bob
1.	Secrets s_0 and s_1. Choose two PKS key pairs (e_0, d_0) and (e_1, d_1).	$\xrightarrow{e_0, e_1}$	
2.		$\xleftarrow{c}$	Choose random key k for symmetric cryptosystem $(\hat{E}, \hat{D})$. Choose random $b \in \{0, 1\}$. Compute $c = E_{e_b}(k)$.
3.	Let $k_i = D_{d_i}(c)$, $i \in \{0, 1\}$. Choose $b' \in \{0, 1\}$. Let $c_i = \hat{E}_{k_i}(s_{i \oplus b'})$, $i \in \{0, 1\}$.	$\xrightarrow{c_0, c_1}$	
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.

Analysis (cont.)

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.

A subtle security problem

Unfortunately, this protocol has a subtle security problem. We claimed that Alice doesn't know Bob's value b after step 2. But why should that be true?

c is the encryption of k using E_{e_0} or E_{e_1}. We would need to know that outputs of $E_{e_0}(k)$ for random k are indistinguishable from outputs of $E_{e_1}(k)$. We have no grounds for believing that.

For example, we could take our favorite PKS and construct a new one where $E'_e(k) = e \cdot E_e(k)$. This is just as secure as the original since e is already known to the adversary. However, the ciphertext here reveals the public key used for encryption.

Another 1-of-2 OT protocol using blinding [1]

	Alice		Bob
1.	Secrets s_0 and s_1. Choose RSA key (n, e, d). Let $y_i = E_e(s_i)$, $i \in \{0, 1\}$.	$\xrightarrow{n, e, y_0, y_1}$	
2.		$\xleftarrow{c}$	Choose random $b \in \{0, 1\}$. Choose random $r \in Z_n^*$. Compute $c = y_b E_e(r) \bmod n$.
3.	Let $c' = D_d(c) \equiv s_b r \pmod n$.	$\xrightarrow{c'}$	
4.			Output $c' r^{-1} \bmod n = s_b$.

[1] This protocol is adapted from notes by David Wagner, U.C. Berkeley, CS276, lecture 29, May 2006.

Analysis

This protocol is much simpler.

- In step 1, Alice sends Bob encryptions of both secrets.
- In step 2, Bob chooses one of Alice's encryptions, blinds it, and returns the result to Alice.
- In step 3, Alice decrypts whatever Bob sends her, which allows Bob to unblind the decryption and recover the secret he chose in step 2.

Alice's other secret is safe assuming semi-honest parties (see lecture 20a) as long as RSA is secure under a limited chosen ciphertext attack (since that is what Alice permits in step 3).

Bob's blinding prevents Alice from knowing which secret he learned.