6.857 Computer and Network Security October 2, 2001

Lecture Notes 8 : Blind Signatures, Commitments, and FOO

Lecturer: Ron Rivest *Scribe: Andrews/Blackwell/Lai/Sieunarine*

Overview

There are three styles of voting proposals: those using blind signatures (FOO), those using homomorphic encryption, and MixNets.

Today we discuss the FOO voting scheme, which utilizes the following three components:

- Blind Signatures
- Commitments
- Anonymous Channels

1 Blind Signatures

We discussed blind signatures briefly last lecture. We will review a bit the use of RSA encryption for this purpose.

With a blind signature, the signer does not know what he is signing, and cannot link it with the resulting signature. This property is called unlinkability. In our voting example, this means that the election officials cannot figure out which voter submitted which ballot. In the case of electronic currency, this means that the bank cannot link the withdrawal of an electronic coin with its expenditure.

Electronic Coin Example: David Chaum's coin protocol

An electronic coin can be considered a signed message by a bank. Three parties are involved in the circulation of an electronic coin: a bank that dispenses the coin, a user who has an account at the bank, and a merchant that accepts electronic currency. The process of circulation (shown in Figure 1) can be simplified to the following:

1. The user withdraws a $1 coin from the bank. The bank sends a signed message that represents the dollar coin, and deletes $1 from the user's account.
2. The user spends the coin, transferring it to the merchant. The merchant can check that the coin is valid by verifying the bank's signature.

[0] May be freely reproduced for educational or personal use.

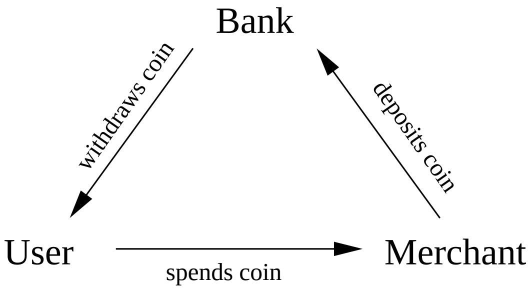


Figure 1: The circulation of an electronic coin.

3. The merchant deposits the coin in the bank.

One of the biggest problems that arises is the double spending of a coin. If the bank does not check that the coin is used only once, it may be used repeatedly by the user or the merchant. If the coins are labelled with identifying information (the user's name, for instance), then this is not so much of a problem, but if we wish to make the coins anonymous, then we need to find another solution. We shall consider a scheme involving blind signatures via RSA, which will allow the coins to not carry identifying information.

Let m denote the message representing a coin, with some redundancy included, perhaps by setting $m = x||h(x)$, where x represents the actual coin data; or perhaps something more elaborate. The redundancy is needed because otherwise the homomorphic properties of RSA allow an adversary to forge coins under this scheme.

Let the bank's public key be (n, e), and the secret key be d. Then the bank's signature is $\sigma = m^d \bmod n$.

The process goes as follows:

1. The user picks a random r to blind the message, then computes $mr^e \bmod n$ and sends it to the bank.
2. The bank signs the message using its private key, and returns $(mr^e)^d \bmod n$ back to the user.
3. Since in RSA $r^{ed} = 1$, the user receives $(mr^e)^d \bmod n = m^d r \bmod n$, so the user simply computes r^{-1} and multiples signed message by it to get the desired result, $m^d \bmod n$.

If the merchant wishes to verify the coin's validity, all she has to do is to raise the coin to the e-th power to get m, and check that it has the desired redundancy.

The bank sees mr^e when the user withdraws the coin and m^d when the merchant deposits the coin; they have no way to correlate the two messages because of the random r chosen by the user. Hence the two message are unlinkable. The FOO voting scheme uses this property to make the ballots cast by the administrators and the ballots seen by the counters unlinkable as well.

Question: How can the bank know that the user has enough balance in the user's account for the amount of the coin the user wants, if the bank cannot see the coin?

Answer: There is a presumption that each coin represents $1, or that the bank's public key is used specifically for $1 coins.

Question: Is there a way to set up different denominations of coins?

Answer: The bank may have multiple public keys, each corresponding to a different denomination. There exists room for further innovation if you want coins that are divisible or otherwise more flexible. (Term project!)

Question: How does one make sure the bank cannot analyze the pattern of messages sent and received to figure out information about them?

Answer: You can assume that there will probably be enough traffic such that the messages are lost in the stream of requests.

2 Commitments

A commitment is like a sealed-bid auction, an encryption, and a one-way function; specifically, it is a two-part protocol consisting of a commit phase and a reveal phase. The commitment protocol is as follows:

1. Alice commits a message m by encrypting and sending it over to Bob.
2. At a later time, Alice reveals the message by sending over the key to decrypt the message. Bob "opens" the commitment and obtains the message m.

There are two properties desirable in a commitment:

Privacy: Bob learns nothing about m having seen $\text{commit}(m)$. There is no way for him to guess m either.

Binding: Alice can only cause one m to be revealed. We want to make sure that she cannot cheat and choose which way Bob can open the message; once she has committed, she can only either stop or let him open the message, and cannot make a different m be revealed.

In the FOO voting scheme, ballots are submitted in committed form, and the voters later reveal the ballots. This prevents partial counting, in which the counters make current tallies available at voting time to try and influence the voters. This also makes the voters go through a second stage after the voting is complete in order to reveal their ballots.

The naïve scheme of encryption actually works pretty well, but we want to makes sure Alice can't find a different key that reveals another message.

The CVP Commitment Protocol

We shall consider the Chaum-van Heijst-Pfitzmann (CVP) commitment protocol. Its security depends on the difficulty of the discrete logarithm problem.

Pick a large prime p, and a generator $g \bmod p$. Then compute $h = g^k \bmod p$, with a secret k that no one should be able to find. (Alternatively, h can be generated randomly such that k is never known.) Also pick r random.

Then $\text{commit}(v) = g^v h^r \bmod p$.

This satisfies both properties.

In yesterday's EECS Colloquium, James Massey distinguished between true (or unconditional) properties and properties that are based on computational assumptions; for example, a property that relies on the difficulty of computing something falls in the latter category. In the case of commitments, the property of privacy is unconditional, while the binding property may not necessarily be unconditional. It may be conditional based on some computationally difficult problem.

To see that the commitment is unconditionally private, consider for the purpose of our argument that h is a generator, or $\gcd(k, p-1) = 1$. Since r is random, even if you think you know v you cannot check for sure – since h is a generator, h^r can be anything in the group.

When the message is revealed, Alice passes (v, r) to Bob.

Can we get a binding property? It depends on the discrete logarithm problem – that, given h, Alice cannot find out what k is.

If Alice wants to cheat, she needs to find v' and r' such that $g^v h^r = g^{v'} h^{r'} \bmod p$.

Question: Are v and r both less than p?

Answer: Yes, we take them mod $p-1$, since they are in the exponents rather than the bases of the exponentiation. Hence $0 < v, r < p-1$. (If we are dealing with ballots, then v should be the hash of the ballot, rather than the entire ballot.)

Then $g^{v-v'} = h^{r'-r} \bmod p$.

As long as $\gcd(p-1, r'-r) = 1$, we can compute the inverse. Raising this to the $(r'-r)^{-1} \bmod (p-1)$ power, we get

$$g^{\frac{v-v'}{r'-r}} = h \bmod p$$

There's a slight gap in our proof: there's no guarantee that $(r'-r)$ has an inverse. However, if it does, then we can raise both sides of the equation by $(r'-r)^{-1}$, and we have found k.

We can fix the gap in our proof by changing the scheme a bit

We know the order of g divides $p-1$. If instead the order of g is prime, $\gcd(r'-r, order(g)) = 1$ automatically.

Make p a large, safe prime such that $p = 2q+1$. Then g is of order q, and k is an unknown such that $0 < k < q$. Now since $\gcd(r'-r, q) = 1$, we can calculate $(r'-r)^{-1} \bmod q$, and let $0 < v < p-1$ and $0 < r < q$. So an inverse exists, and $\frac{v-v'}{r'-r} = k$.

Hence the commitment is binding on Alice.

Question: No one can be trusted to generate h, right? I could generate h, but be on Alice's side, for example.

Answer: We can have a trusted third party generate h.

In practice, we want g and h both to be of order q when coming up with them. Since the divisors of $p-1$ are 2, q, and $2q$, if $g \neq 1$ and $g^q = 1$, we know the order of g is q. We can pick g and h at random, since half of them will be generators.

Question: Can Bob trust Alice to generate h?

Answer: No, someone else must generate h. Even Bob can generate h; it will be okay since privacy is unconditional. If we have multiple commitments going all different ways, however, we will need a third party.

Question: If Bob knows k, can he find v?

Answer: No, since r is chosen randomly.

Question: Are the numbers generated squares?

Answer: Yes, g and h are squares. (So one way of picking g and h would be to pick two "obviously unrelated" squares, e.g. $g = 4$ and $h = 9$.)

Question: Can't Bob change the message after Alice sends it? He knows h and other parameters.

Answer: It depends on what Bob wants to do. He can change the message and then transfer it to a third person in an auction. It won't work for bidding since this scheme is malleable: an adversary Chuck can multiply the commitment by g to get a bid one higher than Alice's. One can disallow bidding with the same r to fix the problem.

Question: How would you know it's the same r?

Answer: You can make it so that Chuck can't hear Alice's opening bid. Perhaps have a recursive commitment—a commitment of a commitment? `:-)`

Question: If you don't have to have the same r, can't you just multiply commit(m) by gh?

Answer: Hmm. Yup, we need a better scheme. The property we need is non-malleability. It may work if you use $g^{\text{hash}(v)} h^r \bmod p$.

There's a difference between auction and voting applications. But what if Bob is married to Alice and wants to cancel her vote?

3 Anonymous Channels

Bob needs to submit his vote anonymously, such that his identity is not transferable across the communications channel. There are several options:

- Channels that offer intrinsic anonymity (radio, satellite rebroadcast)

- Trusted third parties or intermediaries

  Bob sends message to trusted party, who submits message to Counter; or Bob sends message to Agent 1, who sends message to Agent 2, who sends message to Counter.

  A lot of proposals of this sort have been made.

  The third party can strip the identity from the message, but otherwise transmit it faithfully to the intended recipient.

  In MixNets, we have a sequence of these. In FOO, there is only one.

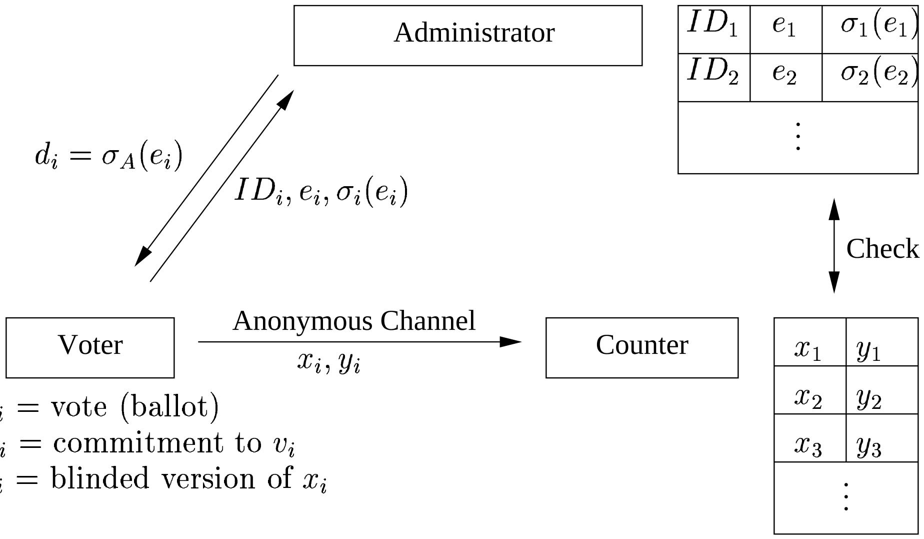
  We also have to worry about timing attacks in which the vote counter keeps track of the voting times and the ballot arrival times.

  Also, if there is only one intermediary for a party with one voter (or other similarly small parties), it's not very anonymous.

We need two anonymous channels in a scheme involving commitments, one for voting (commit) and one for revealing the ballot.

4 The FOO Voting Scheme

The FOO scheme scales well and has several nice properties. Figure 2 illustrates the way FOO works. See handout for a more detailed description of FOO.



ID_1	e_1	$\sigma_1(e_1)$
ID_2	e_2	$\sigma_2(e_2)$
	$\vdots$	

x_1	y_1
x_2	y_2
x_3	y_3
$\vdots$	

Figure 2: A diagram of the FOO voting scheme.

The administrator signs with a blind signature so that the ballots are authenticated, but does not reveal the identity of the voter or what he voted. The counter checks the authentication and also makes sure double counting does not occur.

y_i is the signature of A on x_i.

One can check to make sure all of the votes are on the list and make sure that the number of votes is the same as the number of people.

Claim: the voter's privacy is well-protected. Even if the administrator and counter colluded, the blind signature protects the anonymity of the voter such that there is no correlation between e_i and x_i.

The paper cites more properties, such as not being able to forge another signature from the first one, and the inability to vote twice.

In MIT's implementation of FOO, a big problem is that the administrator can forge votes for voters that did not vote, if he can fake the signature of the voter. This is because the voters don't have public/private keys, and only authenticate through Kerberos. There is therefore no way to contradict a forged vote by the administrators.

Unfortunately, FOO is not a receipt-free scheme, and hence cannot prevent vote-selling. If a voter wants to sell his vote, all he needs to do in order to link e_i and x_i is the blinding factors he used. He can then provide the secret and demonstrate the link to complete the sale.

Question: Can we use WebSIS certificates to upgrade the MIT voting scheme?

Answer: Yes. Term project, anyone?