5.3 BLIND SIGNATURES

An essential feature of digital signature protocols is that the signer knows what he is signing. This is a good idea, except when we want the reverse.

We might want people to sign documents without ever seeing their contents. There are ways that a signer can *almost*, but not exactly, know what he is signing. But first things first.

Completely Blind Signatures

Bob is a notary public. Alice wants him to sign a document, but does not want him to have any idea what he is signing. Bob doesn't care what the document says; he is just certifying that he notarized it at a certain time. He is willing to go along with this.

(1) Alice takes the document and multiplies it by a random value. This random value is called a **blinding factor**.
(2) Alice sends the blinded document to Bob.
(3) Bob signs the blinded document.
(4) Alice divides out the blinding factor, leaving the original document signed by Bob.

This protocol only works if the signature function and multiplication are commutative. If they are not, there are other ways to modify the document other than by multiplying. Some relevant algorithms appear in Section 23.12. For now, assume that the operation is multiplication and all the math works.

Can Bob cheat? Can he collect any information about the document that he is signing? If the blinding factor is truly random and makes the blinded document truly random, he cannot. The blinded document Bob signs in step (2) looks nothing like the document Alice began with. The blinded document with Bob's signature on it in step (3) looks nothing like the signed document at the end of step (4). Even if Bob got his hands on the document, with his signature, after completing the protocol, he cannot prove (to himself or to anyone else) that he signed it in that particular protocol. He knows that his signature is valid. He can, like anyone else, verify his signature. However, there is no way for him to correlate any information he received during the signing protocol with the signed document. If he signed a million documents using this protocol, he would have no way of knowing in which instance he signed which document.

The properties of completely blind signatures are:

1. Bob's signature on the document is valid. The signature is a proof that Bob signed the document. It will convince Bob that he signed the document if it is ever shown to him. It also has all of the other properties of digital signatures discussed in Section 2.6.
2. Bob cannot correlate the signed document with the act of signing the document. Even if he keeps records of every blind signature he makes, he cannot determine when he signed any given document.

Eve, who is
than Bob.

Blind Sign

With the co
"Bob owes Ali
Alice a bag of
many applicat
However, t
taining the u
cut-and-choo
every day, ar
smuggling c
probabilistic
person in ter
Chronic smu
have a 10 pe
penalty for g
nine times.
If the Dep
glers, they h
to search fe
cessful the p
The blind
pile of diffe
then sign th
Think of
the docume
the blindin
nobody can
envelope: V
paper and s
This sce
are secret;
agency's d
of this sig
immunity
just hand
to the age
other han
it. A clev
collects a
signature
Assum
themselv

ıows what he

ıeir contents.
he is signing.

not want him
ːnt says; he is
ong with this.

lue. This ran-

ument signed

ːion are com-
nt other than
now, assume

ːnt that he is
ed document
.ooks nothing
; signature on
ːp (4). Even if
ing the proto-
. that particu-
ːlse, verify his
formation he
signed a mil-
ːing in which

ıroof that Bob
ː document if
of digital sig-

ːning the doc-
ıakes, he can-

Eve, who is in the middle, watching this protocol, has even less information than Bob.

Blind Signatures

With the completely blind signature protocol, Alice can have Bob sign anything: "Bob owes Alice a million dollars," "Bob owes Alice his first-born child," "Bob owes Alice a bag of chocolates." The possibilities are endless. This protocol isn't useful in many applications.

However, there is a way that Bob can know what he is signing, while still maintaining the useful properties of a blind signature. The heart of this protocol is the cut-and-choose technique. Consider this example. Many people enter this country every day, and the Department of Immigration wants to make sure they are not smuggling cocaine. The officials could search everyone, but instead they use a probabilistic solution. They will search one-tenth of the people coming in. One person in ten has his belongings inspected; the other nine get through untouched. Chronic smugglers will get away with their misdeeds most of the time, but they have a 10 percent chance of getting caught. And if the court system is effective, the penalty for getting caught once will more than wipe out the gains from the other nine times.

If the Department of Immigration wants to increase the odds of catching smugglers, they have to search more people. If they want to decrease the odds, they have to search fewer people. By manipulating the probabilities, they control how successful the protocol is in catching smugglers.

The blind signature protocol works in a similar manner. Bob will be given a large pile of different blinded documents. He will **open**, that is examine, all but one and then sign the last.

Think of the blinded document as being in an envelope. The process of blinding the document is putting the document in an envelope and the process of removing the blinding factor is opening the envelope. When the document is in an envelope, nobody can read it. The document is signed by having a piece of carbon paper in the envelope: When the signer signs the envelope, his signature goes through the carbon paper and signs the document as well.

This scenario involves a group of counterintelligence agents. Their identities are secret; not even the counterintelligence agency knows who they are. The agency's director wants to give each agent a signed document stating: "The bearer of this signed document, (insert agent's cover name here), has full diplomatic immunity." Each of the agents has his own list of cover names, so the agency can't just hand out signed documents. The agents do not want to send their cover names to the agency; the enemy might have corrupted the agency's computer. On the other hand, the agency doesn't want to blindly sign any document an agent gives it. A clever agent might substitute a message like: "Agent (name) has retired and collects a million-dollar-a-year pension. Signed, Mr. President." In this case, blind signatures could be useful.

Assume that all the agents have 10 possible cover names, which they have chosen themselves and which no one else knows. Also assume that the agents don't care under which cover name they are going to get diplomatic immunity. Also assume that the agency's computer is the Agency's Large Intelligent Computing Engine, or ALICE, and that our particular agent is the Bogota Operations Branch: BOB.

(1) BOB prepares n documents, each using a different cover name, giving himself diplomatic immunity.
(2) BOB blinds each of these documents with a different blinding factor.
(3) BOB sends the n blinded documents to ALICE.
(4) ALICE chooses $n - 1$ documents at random and asks BOB for the blinding factors for each of those documents.
(5) BOB sends ALICE the appropriate blinding factors.
(6) ALICE opens (i.e., she removes the blinding factor) $n - 1$ documents and makes sure they are correct—and not pension authorizations.
(7) ALICE signs the remaining document and sends it to BOB.
(8) Agent removes the blinding factor and reads his new cover name: "The Crimson Streak." The signed document gives him diplomatic immunity under that name.

This protocol is secure against BOB cheating. For him to cheat, he would have to predict accurately which document ALICE would not examine. The odds of him doing this are 1 in n—not very good. ALICE knows this and feels confident signing a document that she is not able to examine. With this one document, the protocol is the same as the previous completely blinded signature protocol and maintains all of its properties of anonymity.

There is a trick that makes BOB's chance of cheating even smaller. In step (4), ALICE randomly chooses $n/2$ of the documents to challenge, and BOB sends her the appropriate blinding factors in step (5). In step (7), ALICE multiplies together all of the unchallenged documents and signs the mega-document. In step (8), BOB strips off all the blinding factors. ALICE's signature is acceptable only if it is a valid signature of the product of $n/2$ identical documents. To cheat BOB has to be able to guess exactly which subset ALICE will challenge; the odds are much smaller than the odds of guessing which one document ALICE won't challenge.

BOB has another way to cheat. He can generate two different documents, one that ALICE is willing to sign and one that ALICE is not. Then he can find two different blinding factors that transform each document into the same blinded document. That way, if ALICE asks to examine the document, BOB gives her the blinding factor that transforms it into the benign document. If ALICE doesn't ask to see the document and signs it, he uses the blinding factor that transforms it into the malevolent document. While this is theoretically possible, the mathematics of the particular algorithms involved make the odds of BOB's being able to find such a pair negligibly small. In fact, it can be made as small as the odds of Bob being able to produce the signature on an arbitrary message himself. This issue is discussed further in Section 23.12.

Patents

Chaum has p

5.4 IDEN

Alice wants to
from a key ser
his public-key
own compute
Identity-bas
(NIKS) system
network addr
With normal
ates Bob's pul
lic key *is* his
a mail systen
the cryptogra
The syster
If Alice's pri
to get anoth
lusion of dis
A lot of w
of it in Japar
of the propc
in my opin
discussed i
and crypta
1516,1536,
that does
[1546,1547
proposed a
so far is bo

U.S. PAT
4,759,063
4,759,064
4,914,698
4,949,380
4,991,21

ementations
ets simulta-
odifications

hrough sim-
s transfer in
iterations in
ot their mes-
iessage pairs

sly, but says
ise Bob the
Boston's sub-
rotocols are

CHAPTER 6

Esoteric Protocols

6.1 SECURE ELECTIONS

Computerized voting will never be used for general elections unless there is a protocol that both maintains individual privacy and prevents cheating. The ideal protocol has, at the very least, these six requirements:

1. Only authorized voters can vote.
2. No one can vote more than once.
3. No one can determine for whom anyone else voted.
4. No one can duplicate anyone else's vote. (This turns out to be the hardest requirement.)
5. No one can change anyone else's vote without being discovered.
6. Every voter can make sure that his vote has been taken into account in the final tabulation.

Additionally, some voting schemes may have the following requirement:

7. Everyone knows who voted and who didn't.

Before describing the complicated voting protocols with these characteristics, let's look at some simpler protocols.

Simplistic Voting Protocol #1

(1) Each voter encrypts his vote with the public key of a Central Tabulating Facility (CTF).
(2) Each voter sends his vote in to the CTF.
(3) The CTF decrypts the votes, tabulates them, and makes the results public.

This protocol is rife with problems. The CTF has no idea where the votes are from, so it doesn't even know if the votes are coming from eligible voters. It has no idea if eligible voters are voting more than once. On the plus side, no one can change anyone else's vote; but no one would bother trying to modify someone else's vote when it is far easier to vote repeatedly for the result of your choice.

Simplistic Voting Protocol #2

(1) Each voter signs his vote with his private key.
(2) Each voter encrypts his signed vote with the CTF's public key.
(3) Each voter sends his vote to a CTF.
(4) The CTF decrypts the votes, checks the signatures, tabulates the votes, and makes the results public.

This protocol satisfies properties one and two: Only authorized voters can vote and no one can vote more than once—the CTF would record votes received in step (3). Each vote is signed with the voter's private key, so the CTF knows who voted, who didn't, and how often each voter voted. If a vote comes in that isn't signed by an eligible voter, or if a second vote comes in signed by a voter who has already voted, the facility ignores it. No one can change anyone else's vote either, even if they intercept it in step (3), because of the digital signature.

The problem with this protocol is that the signature is attached to the vote; the CTF knows who voted for whom. Encrypting the votes with the CTF's public key prevents anyone from eavesdropping on the protocol and figuring out who voted for whom, but you have to trust the CTF completely. It's analogous to having an election judge staring over your shoulder in the voting booth.

These two examples show how difficult it is to achieve the first three requirements of a secure voting protocol, let alone the others.

Voting with Blind Signatures

We need to somehow dissociate the vote from the voter, while still maintaining authentication. The blind signature protocol does just that.

(1) Each voter generates 10 sets of messages, each set containing a valid vote for each possible outcome (e.g., if the vote is a yes or no question, each set contains two votes, one for "yes" and the other for "no"). Each message also contains a randomly generated identification number, large enough to avoid duplicates with other voters.
(2) Each voter individually blinds all of the messages (see Section 5.3) and sends them, with their blinding factors, to the CTF.
(3) The CTF checks its database to make sure the voter has not submitted his blinded votes for signature previously. It opens nine of the sets to check that they are properly formed. Then it individually signs each message in the set. It sends them back to the voter, storing the name of the voter in its database.

(4) The vot
the CTF
see whic
(5) The vot
CTF's pu
(6) The vot
(7) The CT
a duplic
the vote
number

A malicious vo
ture protocol ens
twice, the CTF w
second vote. If he
this in step (3). M
facility's private
reason.

The cut-and-c
Without that ste
the identificatio

A malicious C
nature protocol
before they are
eventually cast.
voters to confirm

There are still
sent in which
receives votes in
while the CTF
number of signe
covers that the
col, which tries

Voting with

One solution
cheat on its ow

The followin
ers and a separa

(1) Each v
(2) The C
maint
valida

(4) The voter unblinds the messages and is left with a set of votes signed by the CTF. (These votes are signed but unencrypted, so the voter can easily see which vote is "yes" and which is "no.")
(5) The voter chooses one of the votes (ah, democracy) and encrypts it with the CTF's public key.
(6) The voter sends his vote in.
(7) The CTF decrypts the votes, checks the signatures, checks its database for a duplicate identification number, saves the serial number, and tabulates the votes. It publishes the results of the election, along with every serial number and its associated vote.

A malicious voter, call him Mallory, cannot cheat this system. The blind signature protocol ensures that his votes are unique. If he tries to send in the same vote twice, the CTF will notice the duplicate serial number in step (7) and throw out the second vote. If he tries to get multiple votes signed in step (2), the CTF will discover this in step (3). Mallory cannot generate his own votes because he doesn't know the facility's private key. He can't intercept and change other people's votes for the same reason.

The cut-and-choose protocol in step (3) is to ensure that the votes are unique. Without that step, Mallory could create a set of votes that are the same except for the identification number, and have them all validated.

A malicious CTF cannot figure out how individuals voted. Because the blind signature protocol prevents the facility from seeing the serial numbers on the votes before they are cast, the CTF cannot link the blinded vote it signed with the vote eventually cast. Publishing a list of serial numbers and their associated votes allows voters to confirm that their vote was tabulated correctly.

There are still problems. If step (6) is not anonymous and the CTF can record who sent in which vote, then it can figure out who voted for whom. However, if it receives votes in a locked ballot box and then tabulates them later, it cannot. Also, while the CTF may not be able to link votes to individuals, it can generate a large number of signed, valid votes and cheat by submitting those itself. And if Alice discovers that the CTF changed her vote, she has no way to prove it. A similar protocol, which tries to correct these problems, is [1195,1370].

Voting with Two Central Facilities

One solution is to divide the CTF in two. Neither party would have the power to cheat on its own.

The following protocol uses a Central Legitimization Agency (CLA) to certify voters and a separate CTF to count votes [1373].

(1) Each voter sends a message to the CLA asking for a validation number.
(2) The CLA sends the voter back a random validation number. The CLA maintains a list of validation numbers. The CLA also keeps a list of the validation numbers' recipients, in case someone tries to vote twice.

(1) Alice and Bob agree on a random k and an m such that

$$km \equiv e \pmod{n}$$

They should choose the numbers randomly, using a coin-flip protocol to generate k and then computing m. If both k and m are greater than 3, the protocol continues. Otherwise, they choose again.

(2) Alice and Bob generate a random ciphertext, C. Again, they should use a coin-flip protocol.

(3) Alice, using Carol's private key, computes

$$M = C^d \bmod n$$

She then computes

$$X = M^k \bmod n$$

and sends X to Bob.

(4) Bob confirms that $X^m \bmod n = C$. If it does, he believes Alice.

A similar protocol can be used to demonstrate the ability to break a discrete logarithm problem [888].

Zero-Knowledge Proof that n *Is a Blum Integer*

There are no known truly practical zero-knowledge proofs that $n = pq$, where p and q are primes congruent to 3 modulo 4. However, if you allow n to be of the form $p^r q^s$, where r and s are odd, then the properties which make Blum integers useful in cryptography still hold. And there exists a zero-knowledge proof that n is of that form.

Assume Alice knows the factorization of the Blum integer n, where n is of the form previously discussed. Here's how she can prove to Bob that n is of that form [660].

(1) Alice sends Bob a number u which has a Jacobi symbol -1 modulo n.

(2) Alice and Bob jointly agree on random bits: $b_1, b_2, \dots, b_k$.

(3) Alice and Bob jointly agree on random numbers: $x_1, x_2, \dots, x_k$.

(4) For each $i = 1, 2, \dots, k$, Alice sends Bob a square root modulo n, of one of the four numbers: x_i, $-x_i$, ux_i, $-ux_i$. The square root must have the Jacobi symbol b_i.

The odds of Alice successfully cheating are one in 2^k.

23.12 BLIND SIGNATURES

The notion of blind signatures (see Section 5.3) was invented by David Chaum [317,323], who also invented their first implementation [318]. It uses the RSA algorithm.

Bob has a public key, e, a private key, d, and a public modulus, n. Alice wants Bob to sign message m blindly.

(1) Alice chooses a random value, k, between 1 and n. Then she blinds m by computing

$$t = mk^e \bmod n$$

(2) Bob signs t

$$t^d = (mk^e)^d \bmod n$$

(3) Alice unblinds t^d by computing

$$s = t^d/k \bmod n$$

(4) And the result is

$$s = m^d \bmod n$$

This can easily be shown

$$t^d \equiv (mk^e)^d \equiv m^d k \pmod n\text{, so } t^d/k = m^d k/k \equiv m^d \pmod n.$$

Chaum invented a family of more complicated blind signature algorithms in [320,324], called blind unanticipated signatures. These signatures are more complex in construction, but more flexible.

23.13 OBLIVIOUS TRANSFER

In this protocol by Michael Rabin [1286], Alice has a 50 percent chance of sending Bob two primes, p, and q. Alice will not know whether the transfer is successful. (See Section 5.5.) (This protocol can be used to send Bob any message with a 50 percent success rate if p and q reveal an RSA private key.)

(1) Alice sends Bob the product of the two primes: $n = pq$.

(2) Bob chooses a random x less than n, such that x is relatively prime to n. He sends Alice:

$$a = x^2 \bmod n$$

(3) Alice, knowing p and q, computes the four roots of a: x, $n - x$, y, and $n - y$. She chooses one of these roots at random and sends it to Bob.

(4) If Bob receives y or $n - y$, he can compute the greatest common divisor of $x + y$ and n, which is either p or q. Then, of course, $n/p = q$.

If Bob receives x or $n - x$, he can't compute anything.

This protocol may have a weakness: It might be the case that Bob can compute a number a such that given the square root of a you can calculate a factor of n all the time.

23.14 S

This protocol
and Bob toget
to reveal the i
(see Section 6
For this exa
a private key.

(1) Alic

(2) Alic
(3) Bob

D_B i
H
sma
of x

He

and

If t
(4) Bol

(5) Ali
x n
(6) Ali

All the ve
ber appears
knows that
The one
tation befor
(5) and then
step (6).