Lecture Notes, Week 13

1 Kerberos[1]

Kerberos is a widely-used authentication system developed by M.I.T.'s Project Athena. It uses symmetric cryptography with a trusted server to enable secure authentication and key exchange between client nodes. The basic idea is that end-users use long-lived, memorized passwords to get short-lived session keys that aren't worth stealing from insecure desktop computers.

In slightly more detail, each client node has a secret *Kerberos key* that it uses to authenticate itself to the Kerberos server. This key is used only for encrypting session keys. Traffic from server to client is the encrypted using these short-lived session keys. The server stores a database of all client keys, so it must of course be kept highly secure. Ideally, the server is dedicated to this one purpose and does not permit user logins or run any other services.

When Alice wants to talk to Bob, she identifies herself to the server and requests to talk to Bob. The server generates a random session key k for Alice and Bob to use and encrypts it twice, once with Alice's Kerberos key and once with Bob's. It then sends back both encryptions to Alice. Alice decrypts her part and learns k. She sends the other part (called a *ticket*) to Bob, who then decrypts it and also learns k.

If one were only concerned with passive eavesdroppers, this would be enough. All traffic is encrypted with keys not known to Eve, so Eve learns nothing from eavesdropping, assuming of course that the underlying cryptosystem is secure. However, an active eavesdropper could successfully impersonate Alice to Bob by resending the ticket to him at a later time. Although Eve does not know k, she could also resend valid messages encrypted with k that Alice sent earlier and cause Bob to accept them as new messages. This could have serious consequences in a banking application, for example, where Bob is an ATM (automatic teller machine), and Alice's message is the instruction from the bank for the machine to dispense $100.

The real Kerberos protocol is considerably more complicated than the above. First, it has features to protect against various kinds of attacks by active eavesdroppers. Second, it has additional features dictated by practical considerations. These additional features, which we will not discuss in much detail, have to do with making the protocol scalable to large networks and with allowing multiple Kerberos communities to coexist on the same network.

Two versions of Kerberos are in widespread use—versions 4 and 5. While we do not present either protocol in full detail, the simplified protocol we present in the next section relates to the newer version 5.

1.1 Simplified protocol

A simplified version of the Kerberos protocol is shown in Figure 1. In step 1, Alice requests a ticket from Trent (the trusted Kerberos server) to use in talking to Bob. Trent sends back the triple (k, A, L) encrypted twice, once with Alice's private key and once with Bob's. Alice uses the nonce

[1] I thank Don Davis for several helpful comments and corrections to a previous version of this section. Any remaining errors are entirely my own.

Notation:

A	client
B	server
T	trusted Kerberos server
K_{AT}	private key shared by A and T
K_{BT}	private key shared by B and T
k	session key chosen by T
N_A	nonce (random string) chosen by A
T_A	timestamp on A's local clock
L	lifetime (expiration time)
A^*_{subkey}	secret chosen by A
B^*_{subkey}	secret chosen by A

1. Alice sends (A, B, N_A) to Trent (the trusted server).
2. Trent sends two items back to Alice:
   - $E_{K_{AT}}(k, A, L, N_A)$
   - $\text{ticket}_B = E_{K_{BT}}(k, A, L)$
3. Alice decrypts the first item and checks for validity.
   Alice sends two items to Bob:
   - $\text{ticket}_B = E_{K_{BT}}(k, A, L)$
   - $\text{authenticator} = E_k(A, T_A, A^*_{\text{subkey}})$
4. Bob decrypts the ticket using K_{BT} and checks for validity.
   Bob decrypts the authenticator using k, checks validity, and learns A^*_{subkey}.
   Bob sends $E_k(T_A, B^*_{\text{subkey}})$ to Alice.
5. Alice decrypts Bob's message using k.
   Alice checks Bob's message for validity and learns B^*_{subkey}.

Figure 1: Simplified Kerberos protocol.

N_A to match the reply to her request. This prevents her from accepting old messages possibly being replayed by the adversary. She also checks that A is her own ID and that the expiration time L is still in the future.

Alice forwards the ticket to Bob in step 3, and Bob decrypts it in step 4, revealing the session key k. Bob checks that the ticket was issued to Alice and has not yet expired. He then uses k to decrypt the authenticator and checks that it was created by Alice. He also checks that the timestamp T_A is close to the present time on his clock and that Alice has not previously presented him with an authenticator with the same timestamp. (This is also to prevent replay attacks.) At this point, Bob knows that k is the current session key to be used to communicate with Alice.

Bob's transmission to Alice in step 4, and Alice's checks in step 5, let her know that Bob has accepted her credentials and that they have agreed on k as the current session key. The optional subkeys A^*_{subkey} and B^*_{subkey} are additional information that Alice and Bob exchange securely under the protection of k. They can be used for various purposes. For example, Alice and Bob might decide to encrypt their files using $A^*_{\text{subkey}} \oplus B^*_{\text{subkey}}$ as key. Even an untrustworthy Trent that kept a record of all session keys generated could not later decrypt the files unless he was also able to capture the network traffic between Alice and Bob in which A^*_{subkey} and B^*_{subkey} were exchanged.

The purpose of Alice's authenticator is to prevent Eve from using ticket_B to impersonate Alice.

Eve can't generate a valid authenticator because she doesn't know k. The timestamp T_A is used to prevent Eve from intercepting a genuine authenticator from Alice and replaying it at a later time. Bob will only accept T_A once, so unless Eve is sitting between Alice and Bob, Bob will get the real authenticator from Alice before he gets a replay from Eve. Even if Eve is sitting in the middle, the messages that pass by her in steps 3 and 4 are encrypted with keys that she does not possess, so she is neither able to forge such messages nor to learn what they contain.

Note that the protocol assumes that Alice and Bob have well-synchronized clocks. If Bob's clock is very slow, then he could be tricked into accepting an old authenticator. Because clocks are synchronized using a network time synchronization protocol, one possibility for compromising Kerberos is to attack the underlying time synchronization protocol.

Don Davis <dtd@world.std.com> writes:

> "It might interest you that Kerberos v5 no longer depends on a separate time-synch service. In '95, I proposed (with Geer & Ts'o) a way to allow Kerberos to work without strictly-synch'ed clocks, yet without the overhead of replacing all timestamps with challenge-response handshakes, and in fact without requiring any changes to the v5 protocol spec:
>
> http://world.std.com/~dtd/#synch
>
> Our proposal was adopted and implemented in the MIT implementation of Kerberos, and I believe this relaxation of Kerberos's time-synch requirement has contributed substantially to Kerberos's more widespread adoption.
>
> Our paper had two basic ideas:
>
> - Instead of synchronizing clocks, each Kerberos participant should keep track of the clock-skew between its own clock and the kdc's clock, so as to infer the kdc's time-of-day when preparing & interpreting timestamps;
> - Each kerberos user sends a single random challenge in its initial ticket request at login, so as to verify the freshness of the kdc's first ticket's timestamp. the user's kinit client then initializes its record of the clock-skew, by recording the difference between his local login-time and his first ticket's timestamp."

1.2 Practical considerations

As a practical protocol, Kerberos has to be concerned with both security and efficiency. Security considerations dictate that credentials should expire very quickly (on the order of seconds) so that an intercepted ticket cannot be reused later by an attacker to impersonate Alice. Efficiency considerations say the opposite, that credentials should have a relatively long lifetime (on the order of hours), for the more often tickets must be issued, the greater the load on the trusted server. Kerberos solves this dilemma by having two-part credentials. Tickets have a relatively long lifetime and can be used many times. Authenticators have a relatively short lifetime and can be used only once. Only the trusted server can create tickets, but Alice can create a new authenticator each time she wishes to talk to Bob.

Even with tickets having relatively long lifetimes, the need for a single centralized ticket server would prevent Kerberos from being scalable beyond a modest-sized network. To overcome this problem, the full Kerberos protocol splits the trusted server into two parts, an *authentication server (AS)* and a *ticket-granting server (TGS)*. This allows the nodes in a large network to be partitioned into several groups, each with its own server that knows only about the other servers and about the

nodes in its group. Now, for Alice to contact Bob, she first goes to her AS to get a ticket that lets her talk securely to Bob's TGS. She then gets a ticket from Bob's TGS that lets her talk to Bob.

Further details of Kerberos are given in Section 12.4 of your textbook (Wenbo Mao).

2 Secret Ballot Elections

Secret ballot elections are an important real-world problem that combines security issues with significant social and political issues. Florida's problems with punched card voting systems in the 2000 presidential election brought the issue of voting systems to the public's attention. Congress responded with the Help America Vote Act, which provides large amounts of funding to states to upgrade their voting systems. The new systems under consideration are various forms of computerized voting terminals and vote tabulation systems. In addition, some people are calling for internet voting as a way to increase voter participation in elections. In short, there is strong political pressure to use technology to solve real and perceived problems with the existing voting systems. Those who understand the technology and its limitations are rightfully raising red flags warning against the premature adoption of untrusted and untrustworthy equipment into a basically workable system that has evolved over the lifetime of this nation. But I get ahead of myself.

2.1 The Election Problem

Let's begin with the question, "What problem is an election trying to solve?" A naïve answer might be, "To determine the will of the people." When one thinks about it, this answer is not so naïve after all. Many of the properties we demand of a voting system derive pretty directly from this answer. For example, ballots must be secret, for if they are not, then voters will not feel free to express their true opinions, fearing reprisals from those they oppose. The idea that it is important to get as large a voter turnout as possible, that every citizen be allowed to vote, and that ballots must be counted accurately, all derive from the goal of getting an accurate measure of voter sentiment.

Many people however take a narrower view of the question. They postulate that each voter has an internal bit called "voter intent". To them, the goal of a voting system is to determine the exact tally of the voter intent bits. Any failure to count the intent of an individual voter is deemed an offense against that voter. Any documented such failure is grounds for outrage.

This was the basis of the controversy in Florida over the "hanging chad" problem. In a punched card voting system, it is possible for a hole indicating a vote to be only partially punched out. The filling for the hole, called a *chad*, is then left hanging by a few paper fibers. Much debate centered on what the voter's intent was in such a case. Did the voter intend to vote for that candidate but failed to punch the chad completely out of the ballot, or did the voter not intend such a vote, and the chad became partially dislodged because of carelessness on the part of the voter, the election officials, or perhaps a manufacturing defect in the ballot itself? There is really no way to tell. The hanging chads of course introduce ambiguity into the final tally of the election.

I call this *voting system error*, that is, the discrepancy between the quantity to be measured (in this model, "voter intent"), and the actual outcome of the measurement (the official tally). A big motivation behind the Help America Vote Act is to reduce voting system error. While efforts to make voting systems more accurate are certainly laudable, I want to put a little perspective into the issue of voting system error.

First of all, a small error in the election tally is unlikely to affect the outcome of a race. Only when the "will of the people" is nearly equally divided will it matter (as it did in Florida in 2000). In such a case, either outcome can be defended as representing the will of the people, even though

the actual outcome matters greatly to individual candidates and voters.

Secondly, I contend that the model of "voter intent" is an unrealistic and simplistic model of the will of the people. Many people do indeed have well-formulated intent when they enter the voting booth, but many others do not, especially with respect to more obscure offices and races. Ask yourself if you think you could write down the names of the candidates you voted for immediately after leaving the voting booth? Many people simply vote a party ticket and have no idea who the candidates are. When confronted with voting for non-partisan positions, they may decide to abstain, or they may pick a candidate based on vague name-recognition, or just pick randomly. Even for major offices such as president, some voters enter the voting booth still undecided whom to vote for. In the 2000 presidential election, a number of voters were undecided until the last minute whether to vote for Al Gore or Ralph Nader. In the end, they had to make a choice, but there is no reason to believe that had they to vote again that they would make the same choice.

Because of these considerations, I think a better model of voter behavior is a random variable. That is, each voter is modeled as a random process that has a certain probability of voting in a particular way. A dyed-in-the-wool conservative might assign probability 1 to George Bush and probability 0 to Al Gore. For many voters with less extreme views, the probabilities might be more balanced, for example 0.3 for George Bush and 0.7 for Al Gore for a moderate democrat. What is the voter intent in such a scenario? If we run many elections with these two voters, the tally will be 2–0 for George Bush 30% of the time and tied 1–1 the other 70% of the time. Which outcome better reflects voter intent in this case, 2–0 or 1–1? The answer isn't so clear.

One plausible definition of voter intent is the expected value of the tally. In this example, the expected number of votes for Bush is 1.0 + 0.3 = 1.3, and the expected number for Gore is 0.0 + 0.7 = 0.7. One might reasonably say that voter intent favored Bush, but no individual election would ever accurately measure voter intent. If the outcome is 2–0, it overstates Bush's support by 0.7, and if the outcome is 1–1, it understates his support by 0.3. We call this kind of error *statistical voter error*, to distinguish it from the additional error introduced by an imperfect voting system.

The point of making this distinction is that once one is willing to accept the existence of statistical voter error, then the salient question for a voting system is how significant the voting system error is compared to the statistical voter error. If the statistical voter error is already much greater than the error introduced by the voting system, then making the voting system still more accurate will have little positive overall effect on the accuracy of the election.

While on the topic of voting system error, I want to distinguish once again between two kinds of error. There is error such as hanging chads that has a large random component only weakly correlated to voter preference. Some arguments have been made that poorly educated people were more likely to produce hanging chads because they did not understand the importance of completely removing the chad and that those same people have a significant political bias. Nevertheless, this kind of error contrasts dramatically from the kind of adversarial error introduced by various kinds of election fraud—stuffing ballot boxes, voting dead voters, and so forth. Voting systems differ both in their susceptibility to random vote tabulation errors and also in their vulnerability to adversarial error. While random vote tabulation errors can be estimated fairly accurately by standard statistical techniques, adversarial error is not inherently random and cannot be reliably estimated. Whether or not it occurs, and to what degree it occurs, depends, like other security properties, on the motivation, skill, and opportunity of the adversary.

2.2 Shamos's Axioms for Voting Systems

We now consider particular properties that a voting system should have in order to solve the election problem as stated in the last lecture. The following list was formulated by Yale Ph.D. Michael Ian

Shamos, now at Carnegie-Mellon University, and was taken from his paper, "CFP'93 – Electronic Voting – Evaluating the Threat", 1993, which appears on the CPSR web site under URL http://www.cpsr.org/conferences/cfp93/shamos.html.

> **The Six Commandments**
> Democracy is ingrained in the American character and is reflected in its political process from presidential elections down to the most minor of township races. Our passion for fairness and equality has given rise to a set of fundamental requirements for electronic voting systems that are substantially the same from state to state, listed in decreasing order of importance:
>
> I. Thou shalt keep each voter's choices an inviolable secret.
>
> II. Thou shalt allow each eligible voter to vote only once, and only for those offices for which she is authorized to cast a vote [2].
>
> III. Thou shalt not permit tampering with thy voting system, nor the exchange of gold for votes.
>
> IV. Thou shalt report all votes accurately.
>
> V. Thy voting system shall remain operable throughout each election.
>
> VI. Thou shalt keep an audit trail to detect sins against Commandments II–IV, but thy audit trail shall not violate Commandment I.

[2] Recall that women now constitute a majority of registered voters in the United States.

These are a tough set of requirements to meet in practice. Let's see how our existing voting systems work.

1. Votes are kept secret by being dissociated from the voter. With absentee ballots, the dissociation comes only when the outer envelope is opened at the election official's office. With paper ballots, it occurs when the ballot is dropped into the ballot box. With some voting machines, the identity of the voter is never entered into the machine, but this is clearly not true of internet voting, nor would it be true of computerized voting machines that attempted to integrate the voter authentication process with the actual balloting.

2. Voter registration and poll watchers attempt to enforce the principle of one man, one vote. Voter registration is restricted to legally authorized persons. Poll watchers attempt to match voters presenting themselves in person with names on the registration list. The name is crossed off, preventing the same person from voting multiple times. Neither the voter registration nor the voter authentication processes are particularly secure. Registrars often accept a person's signed statement that he or she is eligible to vote without further investigation or documentation. Poll watchers may accept a person's declared identity, or they may require the presentation of some sort if identification card. Little is done to prevent the same person from registering and voting in multiple states, even under their own name, nor is there much to prevent the same person from registering and voting multiple times under different names. Such voter fraud is of course a serious crime, so the threat of punishment and the limited personal gain from successful fraud work together as a strong disincentive to cheat.

3. Tampering with the voting system is discouraged by having as much as possible of the election process take place in the open, subject to the scrutiny of representatives of both political

parties and other interested groups. Large-scale tampering is hampered by the decentralized nature of the election process. Each election district generally runs its own election and counts its own votes.

Vote-selling is discouraged by the inability of a voter to provide evidence of how she voted, so a party interested in buying votes cannot know whether or not a voter complied with their wishes.

4. The tallying of the votes is a two-step process. The ballots are counted publicly by the precincts. Only the tabulation of the precinct totals is centralized. The latter computation is easily verified through independent computation by news agencies and other interested groups.

5. Conscientious election officials spend months planning an election to ensure that needed personnel, equipment, and supplies are available throughout election day. Obviously, individual precincts can and do experience disruptions—power can fail, election workers can be taken suddenly ill, voting machines can malfunction, and so forth, but decentralization again works in favor of the overall election process. A tornado that tears the roof off of one polling place (and yes, that could happen) is unlikely to affect a polling place in the next county or next state.

6. Because of the need to maintain secrecy of the vote, the audit trail is split into two pieces. On the one hand, there are records of who voted (or at least which names on the registration list are marked as having voted). On the other, there are the actual ballots, if paper ballots are being used. The number of ballots cast can be compared with the number of people who voted and should be equal except for the possibility of a voter walking away without depositing the ballot in the ballot box. But the audit trail does not enable one after the fact to verify that the ballots in the ballot box were in fact the ones cast by the voters whose names were crossed off the registration list.

Mechanical voting machines and many direct-recording computerized voting terminals lack the ability to provide a *voter-verified* audit trail. With such machines, the votes are stored internally in the machine until the polls close, at which point election workers read out the totals. Here, there is no way for a voter to know that her vote was counted correctly. Obviously, the total votes cast on a machine can be compared with the number of people who voted on that machine, but there is no way to connect the votes recorded with the votes cast by the voter. With mechanical machines, the linkages between levers and counters are relatively simple and visible. One can have some degree of confidence that the machine is working as intended just through a visual inspection of its innards and some simple testing of its operation.

With computerized voting terminals, all such transparency is lost. One cannot look at a computer and know what is going on inside. Testing is insufficient since computers can easily be programmed to respond normally except when given some kind of coded signal, which might be nothing more than a ballot being cast for a particular unlikely collection of candidates. Such a machine might perform flawlessly during testing, yet corrupt the vote on election day after a malicious voter, in the privacy of the voting booth, cast the coded ballot that triggered the embedded malware. Some confidence might be gained by an examination of the source code, but most voting machine companies consider their code to be proprietary and require through their contracts that it be kept confidential.

2.3 Internet Voting

An interesting question, especially for a cryptography class, is how one might carry out elections completely electronically, on the internet for example. This is not just a hypothetical. The Department of Defense planned to allow troops overseas vote via the internet in the 2004 presidential election. This idea was later scrapped after much protest. (See www.theocracywatch.org/pentegon_cancels_electr_voting.htm.)

Internet elections pose serious challenges to almost all of Shamos's commandments. How can an electronic ballot be both private and countable? How can voting be restricted to authorized voters in a way that prevents the tabulating authority from being able to match voters to ballots? How does one prevent tampering with the software used to implement the system, both on the user's computer and also at the central authority? How does one prevent a voter from selling her voter credentials to a third party? How does one prevent a voter from voluntarily letting a third party cast her ballot on her behalf? How can one decentralize an internet voting system so as to obtain some of the robustness that our current decentralized system enjoys? How can one prevent targeted denial-of-service attacks that could disable internet voting for certain groups of voters on election day? What kinds of electronic audit trails are possible, and how can they be verified?

Several election protocols have been devised that attempt to address some of these questions. While none is entirely satisfactory from a practical point of view, they do employ some clever cryptographic ideas that might eventually be useful.

2.3.1 Voting with two central facilities

A simple voting solution with a trusted central authority (CA) simply has each voter encrypt, using the CA's public key, a ballot containing voter identification information along with a marked ballot. The CA decrypts each ballot, checks the voter against the registration list, verifies that the voter has not already voted, and counts the ballot. This is essentially what happens with absentee ballots, but it offers no real guarantee of privacy.

One way to enhance privacy is to split the CA into two parts: The central legitimizing authority (CLA) is responsible for voter authentication, and the central tabulating authority (CTA) is responsible for tabulating the votes. Here's how the scheme might work. The voter requests a ballot from the CLA. The CLA authenticates the voter, checks the registration list, assigns the voter a random voter ID, and returns a ballot to the voter along with the voter ID. The voter marks the ballot, encrypts it along with the ID using the CTA's public key, and sends it off to the CTA. The CTA decrypts the message and makes a list of ballots with attached voter ID's. The CTA does not know the voter identity corresponding to a given ID, so the CTA does not know which voter cast which ballot. When the polls close, the CLA sends a list of valid ID's to the CTA. The CTA then discards any ballots with ID's not on the official list and tallies the remainder.

This system maintains privacy of votes as long as either the CLA or the CTA is honest. An honest CLA never reveals the association between voters and voter ID's. An honest CTA never reveals the association between voter ID's and votes. Of course, if they collaborate, they can learn how everyone voted. It also relies on trust that the CTA tabulates votes correctly. There is nothing in this protocol to prevent the CTA from simply replacing the real ballots with ones that it creates.

Another problem with this scheme has to do with voter disenfranchisement. A dishonest CLA could prevent the votes of selected voters from being counted by simply dropping their ID's from the official list sent to the CTA. The voter wouldn't know this had been done, and the CTA would only know that they received a number of ballots with invalid ID's. A possible fix would be for the CTA to make the list of valid voter ID's public so that each voter could verify that their own ID was

included on the list. As long as this was done after the polls closed, it would not permit one voter to steal another voter's ID.

Yet another problem is that the CTA might learn the voter's true identity when the ballot is submitted. This could be prevented by submitting the ballot using an anonymizing service, or by having the ballot go back to the CLA who would then forward it to the CTA. Remember, the ballot is encrypted with the CTA's public encryption key, so the CLA cannot read the ballot.

2.3.2 Peralta's scheme

René Peralta has proposed a simple electronic ballot based on blind signatures. A *blind signature* scheme provides a way to obtain a valid signature on a document that the signer cannot read. This is exactly what we want for voting. Imagine a situation where a voter marks a ballot and seals it inside an envelope lined with carbon paper. She then take the envelope with the enclosed ballot to the CLA, who signs the envelope without opening it. The CLA's signature on the envelope is transfered to the ballot by the carbon paper. Later, the voter removes the validated ballot from the envelope and places it in the ballot box. Blind signatures are an electronic analog of this scheme.

In Peralta's scheme, a ballot has four fields:

- election ID
- random voter ID
- vote
- signature of election authority

The voter prepares a ballot encrypted by the CTA's public key, identifies herself to the CLA, gets the CLA to sign the encrypted ballot, then submits the ballot (via an anonymous channel) to the CTA for counting. The CTA checks the CLA's signature, decrypts the ballot, and counts it.

The clever part is how the voter obtains the CLA's signature without revealing her vote. Let x be the concatenation of the first three fields of the ballot. The voter needs to obtain the CLA's signature $S_{\mathrm{CLA}}(x)$ in order have a valid signed ballot for submission to the CTA. Here's how she does this assuming the CLA is using RSA signatures with private signing key (d, n) and public verification key (e, n):

1. Voter chooses random number $r \in Z_n$. r is called a *blinding factor*.
2. Voter computes $z = xr^e \bmod n$ and sends z to the CLA along with the voter's name and other authentication information.
3. The CLA authenticates the voter and issues $S_{\mathrm{CLA}}(z) = z^d \bmod n$ to the voter. Note that $z^d \bmod n = x^d r^{ed} = x^d r \bmod n$.
4. Voter calculates $S_{\mathrm{CLA}}(x) = x^d \bmod n = z^d r^{-1} \bmod n = S_{\mathrm{CLA}}(z) r^{-1} \bmod n$.
5. Voter submits the signed ballot $(x, S_{\mathrm{CLA}}(x))$ to the CTA via an anonymous channel.

Note that vote privacy is ensured even if the CLA and CTA conspire to reveal the votes. This is because the number z is a random number that is largely uncorrelated with the ballot x. Thus, even if the CLA retains a list of all the z that it has signed and who they belong to, it has no way of matching up the z's with the x's that are later received.