Chapter 10

Distributed-Applications Security

Contents

10.1 Database Security

Databases are often crucial elements of internal networks and web applications. Because databases play such an important role in storing large amounts of potentially valuable information, they are often the target of attacks by malicious parties seeking to gain access to this data. Thus, an important element of computer security involves protecting the confidentiality, integrity, and availability of information stored in databases.

In addition, databases often contain sensitive information that may reveal details about individuals as well, so another security concern with respect to databases is privacy. (See Figure 10.1.)

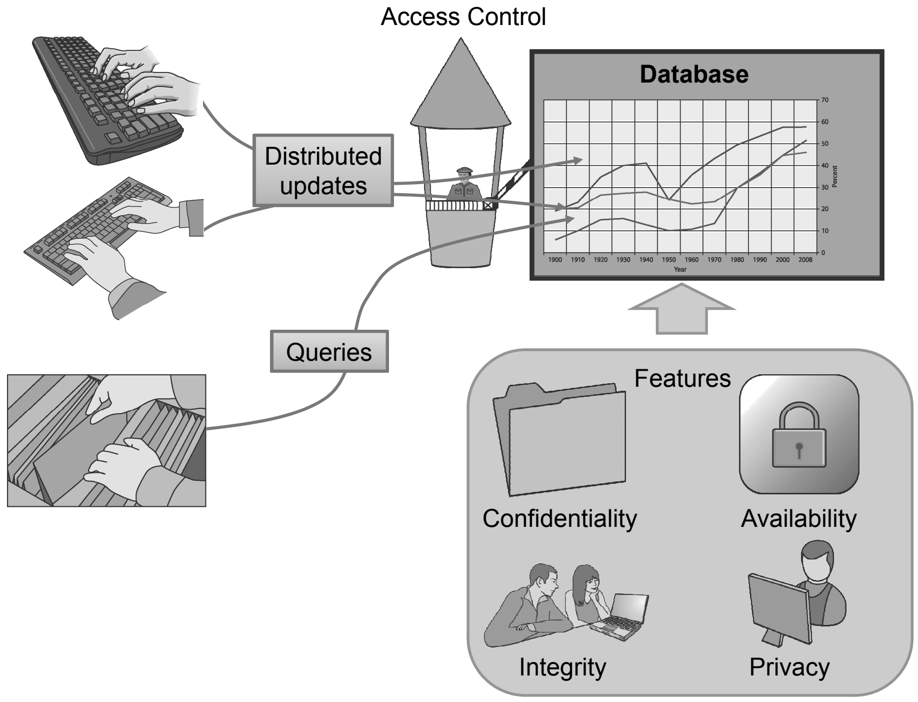


Figure 10.1: Databases must deal with distributed updates and queries, while supporting confidentiality, availability, integrity, and privacy. Doing this requires strong access control as well as mechanisms for detecting and recovering from errors.

10.1.1 Tables and Queries

A very common way to store information is to use a ***relational database***. In this approach, information is organized into a collection of ***tables***. Each row of a table is a ***record*** that stores related information about some entity and each column is associated with an ***attribute*** that the entity can possess. An example table of a relational database is shown in Figure 10.2.

Num	**Name**	**Inaugural_Age**	**Age_at_Death**
1	George Washington	57.2	67.8
2	John Adams	61.3	90.7
3	Thomas Jefferson	57.9	83.2
4	James Madison	58.0	85.3
5	James Monroe	58.8	73.2
6	John Quincy Adams	57.6	80.6
7	Andrew Jackson	62.0	78.2
⋮	⋮	⋮	⋮
26	Theodore Roosevelt	42.9	60.2
27	William Howard Taft	51.5	72.5
28	Woodrow Wilson	56.2	67.1
29	Warren G. Harding	55.3	57.7
30	Calvin Coolidge	51.1	60.5
31	Herbert Hoover	54.6	90.2
32	Franklin D. Roosevelt	51.1	63.2
33	Harry S. Truman	60.9	88.6
34	Dwight D. Eisenhower	62.3	78.5
35	John F. Kennedy	43.6	46.5
36	Lyndon B. Johnson	55.2	64.4
37	Richard Nixon	56.0	81.3
38	Gerald Ford	61.0	93.5
39	Jimmy Carter	52.3	
40	Ronald Reagan	70.0	93.3
41	George H.W. Bush	64.6	
42	Bill Clinton	46.4	
43	George W. Bush	54.5	
44	Barack Obama	47.5	

Figure 10.2: A relational database table, `Presidents`, storing data about U.S. presidents. This table has 44 records (rows) and 4 attributes (columns), the last two of which are numeric values (expressing years) or null values.

SQL Queries

As mentioned in Section 7.3.3, most databases use a language known as ***SQL*** (***Structured Query Language***) to support queries and updates, using commands that include the following:

- SELECT: to express queries
- INSERT: to create new records
- UPDATE: to alter existing data
- DELETE: to delete existing records
- Conditional statements using WHERE, and basic boolean operations such as AND and OR: to identify records based on certain conditions
- UNION: to combine the results of multiple queries into a single result

These commands can be combined to produce queries that extract data, or updates that make changes to the database. Suppose, for example, we were to issue the following query on the table of Figure 10.2.:

```
SELECT * FROM Presidents WHERE Inaugural_Age < 50
```

This query is designed to find and return all the U.S. presidents who were younger than 50 when they were inaugurated. The star symbol (*) specifies to return all the attributes of the resulting records. This query would return the following table, which consists of a subset of the records of table `Presidents`:

Num	Name	Inaugural_Age	Age_at_Death
11	James K. Polk	49.3	53.6
14	Franklin Pierce	48.3	64.9
18	Ulysses S. Grant	46.9	63.2
20	James A. Garfield	49.3	49.8
22	Grover Cleveland	48.0	71.3
26	Theodore Roosevelt	42.9	60.2
35	John F. Kennedy	43.6	46.5
42	Bill Clinton	46.4	
44	Barack Obama	47.5	

More complex queries are also possible, such as one to find all U.S. presidents who were less than 50 when they took office and died during their first term:

```
SELECT * FROM Presidents WHERE (Inaugural_Age < 50)
      AND (Age_at_Death - Inaugural_Age < 4.0)
```

This query would return the following set of records:

Num	Name	Inaugural_Age	Age_at_Death
20	James A. Garfield	49.3	49.8
35	John F. Kennedy	43.6	46.5

10.1.2 Updates and the Two-Phase Commit Protocol

In addition to queries that extract information from a database, authorized users can also update the contents of a database using SQL commands. For example, the following update operation would delete all of those records from the `Presidents` table that correspond to U.S. presidents who were less than 50 years old when they were inaugurated:

```
DELETE FROM Presidents WHERE Inaugural_Age < 50
```

In addition, the following update operation would add a new record to the `Presidents` table:

```
INSERT INTO Presidents
VALUES (45, 'Arnold Schwarzenegger', 65.5, NULL)
```

Database updates can be more fine-grained than just inserting and deleting entire records, however. We can also alter the contents of individual attribute values in specific records. For example, continuing our running example, one would imagine that, prior to December 26, 2006, the `Presidents` table contained the following record:

Num	Name	Inaugural_Age	Age_at_Death
38	Gerald Ford	61.0	

After December 26, 2006, however, one would expect that an agent who is authorized to make changes to this table would have issued a command like the following:

```
UPDATE Presidents
SET Age_at_Death=93.5
WHERE Name='Gerald Ford'
```

This command would have updated just a single attribute value—the `Age_at_Death` field—for a single record—the one that has a name field that matches the string `'Gerald Ford'`—resulting in the record above to change as follows:

Num	Name	Inaugural_Age	Age_at_Death
38	Gerald Ford	61.0	93.5

Ideally, a database would allow for multiple authorized agents to be updating and querying a database at the same time. All of these operations would be logged to an audit file, to provide a lasting record of the types of information that were extracted from the database and a history of the changes that were made to that database as well.

Two-Phase Commit

One of the big challenges of allowing for multiple agents to be updating a database at the same time in a distributed fashion on a network is that update operations can conflict. For example, if Alice wants to delete a record and Bob wants to change one of the attribute values for that same record at the same time, then there is a problem. In addition, even if multiple simultaneous updates don't conflict, there is a chance that there could be a computer or network failure during one of these updates so that it doesn't completely finish the update. Such a failure could leave the database in an inconsistent state, which could even make it unusable.

To cope with with these consistency and reliability issues, most databases employ a protocol called ***two-phase commit*** for performing updates. The sequence of operations proceeds along two phases:

1. The first phase is a ***request phase***, in which all the parts of the database that need to change as a result of this update are identified and flagged as being intended for this change. The result of this phase is either that it completes successfully, and every change requested is available and now flagged to be changed, or it aborts, because it couldn't flag all the parts it wanted (say, because someone else already flagged it) or because of a network or system failure. If the first phase aborts, then all its requested changes are reset, which is always possible, because no permanent changes have been made yet. If the first phase completes successfully, then the protocol continues to the second phase.

2. The second phase is the ***commit phase***, in which the database locks itself into other changes and performs the sequence of changes that were identified in the request phase. If it completes successfully, then it clears all the flags identifying requested changes and it releases the lock on the database. If, on the other hand, this operation fails, then it rolls back, that is, reverses, all the changes made back to the state the database was in just after completing the first phase.

This two-phase commit protocol is therefore a feature that a database can use to help achieve both integrity and availability. It supports integrity, because the database is always either in a consistent state or it can be rolled back to consistent state. This protocol supports availability, as well, because the database is never put into a state of internal inconsistency that would cause the database management system to crash.

10.1.3 Database Access Control

Databases employ several security measures to prevent attacks, protect sensitive information, and establish a security model that minimizes the impact of database compromise. While implementation details depend on the database, most databases provide a system of access control that allows administrators to dictate exactly what certain users and groups are permitted to do in relation to that database.

For instance, many systems implement an access-control list (ACL) scheme similar to those used by operating systems. A simple access-control system might allow a web application to perform search queries on the data and insert new records, for example, but not create or remove tables or execute system commands via the database. More complicated sets of rules may also be used to define different sets of permissions for multiple users. For example, a database that includes tables of student records and university employment records might allow faculty members to insert and update grades for students, but not allow them to make changes to their own employment records. A dean, on the other hand, might be granted rights to make additions and modifications to both student and employee records.

In general, being able to define access-control permissions for the various users of a database can be a significant benefit, helping to minimize damage from insider attacks, such as information leakage by overly curious employees or students who try to change the grades in their transcripts. A proper set of access controls should implement a ***least-privilege principle*** (Section 1.1.4), so that each user has the necessary rights to perform their required tasks, but no rights beyond that.

Properly defined access permissions can also be a critical preventive measure for database compromise in the event of an intrusion. For example, consider a database that stores information for two sections of a subscription-based news web site, articles and photos in one section, and financial records about customers in the other section (e.g., credit card numbers). In this case, the database and web application should be configured so that each portion of the application only has access to the necessary information for that portion. With this safety measure in place, if the unprivileged news section of the web site is compromised, the attacker would be unable to access sensitive customer information. Thus, by designing access privileges using the concepts of ***least privilege*** and ***separation of privilege***, damage from intrusions can be minimized.

Access Control Using SQL

SQL defines an access control framework that is commonly used for defining database privileges. When a table is created, the owner of the table has the sole rights to perform operations on that table. The owner can then ***grant*** privileges to other users, which is known as ***privilege delegation***. These privileges may be broad, such as the ability to do anything to a particular table, or fine-grained, such as the ability to perform only SELECT queries on certain columns. For example, the owner of a table may issue the following SQL command to give Alice the ability to search through table `employees`:

```
GRANT SELECT ON employees TO Alice;
```

Other permissions that can be provided using the GRANT keyword include DELETE, INSERT, and UPDATE. In addition, to grant all available rights one can use the ALL keyword.

Permissions can be granted to individuals or to everyone (using the PUBLIC keyword). In addition, permissions can be granted to roles, allowing for role-based access control for a database.

In addition, the owner of a table can create a virtual subset of the data known as a ***view***, which can then be accessed by other users. For example, the owner of a table may wish to allow a user, Alice, to update only her own information. This can be accomplished by creating a view of the total dataset that only includes Alice's data, and granting update access on this view to Alice.

Privilege Delegation and Revocation

In addition to being able to grant certain privileges to other users, table owners can also allow other users to grant privileges for those tables, which is known as ***policy authority delegation***. Specifically, when granting a privilege to a user as in the above examples, the grantor can include the clause WITH GRANT OPTION to give the recipient the ability to further delegate that privilege. For example, an administrator might create a view for Alice and give her permission to delegate SELECT permissions on that view to other users as follows:

```
CREATE VIEW employees_alice AS
  SELECT * FROM employees
  WHERE name = 'Alice';
GRANT SELECT ON employees_alice TO Alice WITH GRANT OPTION;
```

Visualizing Privilege Propagation and Revocation

The propagation of privileges in a database can be visualized using a diagram, where nodes represent users and directed edges represent granted privileges. If Alice grants a set of rights, A, to Bob, then we draw a directed edge labeled with A from Alice to Bob. A user, Alice, who has granted privileges to another, Bob, can opt to revoke those privileges at a later time, which would be visualized by deleting or relabeling the edge from Alice to Bob. A command that could perform such a revocation is as follows:

```
REVOKE SELECT ON employees FROM Bob;
```

This command should result in the revocation of all SELECT privileges for Alice as well as all the people to which she had delegated this privilege. For example, consider the case where a user Alice grants a set of privileges to Bob, who in turn grants those privileges to Carol. If Alice revokes these privileges from Bob, then the entire path of delegated propagation should be followed so that both Bob and Carol have this set of privileges revoked. This revocation scenario becomes a bit more complicated when multiple users have granted Bob overlapping sets of privileges, and only one user revokes these privileges. Intuitively, Bob should retain the complete set of privileges granted by the user who did not issue a revocation, and any grantees who were granted privileges by Bob should only have those privileges revoked if Bob was authorized to perform this granting by a different, unrevoked set of privileges. (See Figure 10.3.)

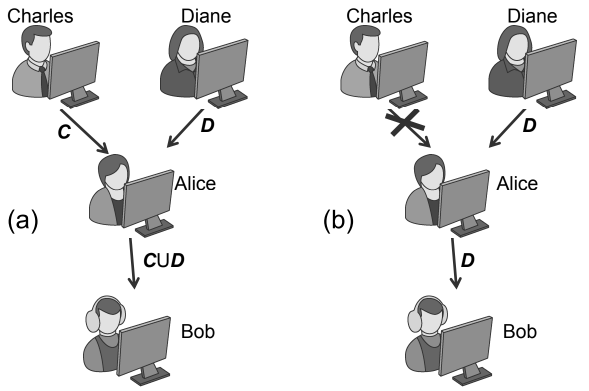


Figure 10.3: How database privileges can be visualized with a directed diagram: (a) First, two administrators, Charles and Diane, each grant Alice two sets of privileges, C and D, after which Alice grants those privileges to Bob, giving him the set of rights in the union, $C \cup D$. (b) If Charles subsequently revokes the set of privileges, C, he granted to Alice, then the privileges Bob inherited indirectly from Charles, through Alice, should also be revoked, leaving Bob with just the privileges in D.

Propagating Privilege Revocation

Implementing correctly privilege delegation and revocation requires some additional overhead. The formal meaning of privilege revocation is that the privileges given to users should be the same as if the revoked privilege had never been granted. Recomputing all privileges for each user from scratch by replaying all the GRANT statements ever issued, except the revoked one, is computationally very onerous.

The technique described below allows to efficiently identify the impact of revocation statements by maintaining a ***time stamp*** for each privilege granting action. Namely, the database keeps a table, denoted `grants`, whose attributes are the grantor, grantee, privilege, and time stamp of the grant. A user holds a certain privilege, P, if table `grants` has at least one record that contains P. Suppose, for example, that table `grants` has the following entries:

Grantor	**Grantee**	**Privilege**	**Timestamp**
Alice	Carol	P	1
Bob	Carol	P	2
Carol	David	P	3

Next, at time 5, Alice revokes her grant of privilege P to Carol. As a consequence, the first record in table `grants` is removed. However, Carol still has privilege P since it has been granted to her also by Bob. But how about the grant of privilege P that Carol has made to David at time 3? Should this record be removed, causing David to lose privilege P? The answer is no because when this grant was done by Carol, she had a previously issued (at time 2) grant for P from Bob. Thus, even in the absence of the grant from Alice, Carol could have made a valid grant to David. Suppose instead that the grant from Bob to Carol had been made at time 4, In this case, Carol could not have been made a valid grant to David at time 3. Thus, the associated record should be removed from table `grants`. The algorithm for propagating privilege revocation is formally expressed below in pseudocode.

```
REVOKE( record $X$)
 1  let $X = (A, B, P, t)$
 2  delete record $X$ from grants
 3  $t^* \leftarrow current\_time$
 4  for each record $R$ such that $R.grantee = B$ and $R.privilege = P$
 5      do if $R.timestamp < t^*$
 6            then $t^* \leftarrow R.timestamp$
 7  // $t^*$ is the earliest time stamp of a grant of $P$ to $B$
 8  for each record $R$ such that $R.grantor = B$ and $R.privilege = P$
 9      do if $R.timestamp < t^*$
10            then REVOKE($R$)
```

10.1.4 Sensitive Data

In addition to ensuring that databases have appropriate access-control measures in place, care must be taken to guarantee that sensitive data is stored in a way that protects the privacy of users and any confidentiality requirements for sensitive data.

Using Cryptography

If information being stored in a database has confidentiality requirements, then it should not be stored in plaintext, but should instead be stored as the output of a cryptographic function. As an example, consider a web site that stores passwords for user accounts in a database. Recalling the password-based authentication methods covered in Section 3.3.2, these passwords should never be stored in plaintext, or an intrusion could result in the compromise of every user account. Instead, a cryptographic hash of each password and its salt should be stored. When a user attempts to log in, the password provided by the user and the salt stored in the database would be hashed and compared against the stored hash value. This way, if an attacker compromised the database, they would acquire a list of hashes, from which the actual passwords could not be recovered unless a dictionary or brute-force attack proved successful.

As another example, confidential files kept in a database should be stored in encrypted form, where the decryption key should be known by authorized users but not stored in the database itself. However, standard encryption methods prevent searching for files by providing keywords.

Privacy Protection

Besides measures designed to protect the confidentiality of sensitive user information, database owners should be careful to consider the privacy impacts of publishing or granting access to sensitive information. If a database is to be released to the public, say, to be used for research purposes, then all identifying information, such as names, addresses, Social Security numbers, employee numbers, and student numbers, should be removed or changed to ***masking values***, which are nondescript values that lack all identifying information. For example, a database of employees might be made public after each employee name is replaced with a unique ID, like `id001`, `id002`, `id003`, and so on.

Inference Attacks

Even if identifying information is removed or masked out, it may still be possible to use the database in conjunction with additional information available to the attacker to learn more about the underlying data. This is referred to as an ***inference attack***. As an example, consider a database of employee records, whose attributes are name, gender, ID number, and salary. Suppose a party is granted access to a sanitized version of the table, where the name attribute is removed, for the purpose of creating statistics on salary by gender. Another party may have a list of pairings associating ID numbers to names for a reporting task. If these two parties were to communicate, they could easily infer the salary of each employee, despite the intent of the database owner. In general, when granting access to modified versions of a database, administrators should consider whether collusion among grantees can allow them to gain unauthorized information.

Protecting Databases Against Inference Attacks

To protect a database from inference attacks, the following techniques can be used prior to making the database public. (See Figure 10.4.)

- ***Cell suppression***. In using this technique, some of the cells in a database are removed and left blank in the published version. The goal is to suppress the critical cells that could be used in an inference attack to determine sensitive implications for individuals.

- ***Generalization***. In using this technique, some values in a published database are replaced with more general values. For example, a date of birth, like "June 2, 1983," could be replaced with a range of years, like "1980–1984;" or a zip code, like "92697-3435," might be changed to "926xx-xxxx." The goal is to generalize critical values so that they become mixed with other values, to make inference attacks less feasible.

- ***Noise addition***. In using this technique, values in a published database have random values added to them, so that the noise across all records for the same attribute averages out to zero. For example, an age value could have a random value in the range from -5 to 5 added to it. The goal is to obscure individual values while leaving the average value unchanged.

Of course, all of these techniques make the information in a published database less specific, which might be required by some regulations, such as the requirement of the U.S. Census Bureau to never publish information that can be directly traced to any individual U.S. citizen.

Num	Age1	Age2
11	49.3	53.6
18	46.9	63.2
20	49.3	49.8
35	43.6	46.5
42	46.4	
44	47.5	

(a)

Num	Age1	Age2
11	49.3	
18	46.9	63.2
20	49.3	
35		
42	46.4	
44	47.5	

(b)

Num	Age1	Age2
11	45–50	50–60
18	45–50	60–75
20	45–50	45–50
35	40–45	45–50
42	45–50	
44	45–50	

(c)

Num	Age1	Age2
11	47.7	55.2
18	49.2	64.3
20	51.6	52.8
35	42.3	47.3
42	47.1	
44	48.0	

(d)

Figure 10.4: Obfuscation techniques for protecting the privacy of individuals included in a public database: (a) A table with individual names removed. (b) A table anonymized using cell suppression. (c) A table anonymized using generalization. (d) A table anonymized using noise.

Given the obfuscation techniques above, there is clearly a question of how far to go in applying them to provide a sufficient amount of privacy protection. In the extreme, we could "blur" the data so much that it is completely useless, being little more than a database of random noise and blank cells. This would protect data privacy, but it would also be completely useless. Thus, we need to apply the obfuscation techniques above in conjunction with some rule for deciding when data has been sufficiently obscured. Unfortunately, there is, as of yet, no widely accepted standard for deciding when information in a public database has been sufficiently obscured. Nevertheless, proposed definitions include the following:

- ***k*-anonymization**. In this standard, a database is considered sufficiently anonymized if any possible SELECT query would return at least k records, where k is a large enough threshold of disclosure tolerance.
- ***Differential privacy*** In this standard, a database is considered sufficiently anonymized if, for any record R in the database, the probability, p, for some sensitive property, P, with R being in the database, and the probability, p', for the property, P, with R not being in the database, differ by at most ϵ, where ϵ is a small enough threshold of information leak tolerance.

Of course, both of these properties provide a quantifiable level of privacy.

10.2 Email Security

Electronic mail is one of the most widely used Internet applications. Indeed, the ability to send messages and files to specific groups or individuals via the Internet is such a powerful tool that it has changed the way people communicate in general. Because of this wide and ubiquitous usage, addressing the security of email requires that we discuss several classic security issues, including authentication, integrity, and confidentiality. We study these issues in this section by briefly explaining how email works, and then examining technologies that accomplish various security goals for email. Finally, we will take a look at an important security problem related to email—spam.

10.2.1 How Email Works

Today's email systems make use of several protocols to deliver messages. To handle the sending of messages from a client's machine to a recipient's mail server, the ***Simple Mail Transfer Protocol*** (***SMTP***) is used. SMTP is a simple text-based, application-layer protocol that uses TCP to facilitate a "conversation" between a client wishing to send mail and an appropriate receiving server. In the SMTP model, the client is referred to as the ***Mail User Agent*** (***MUA***). The MUA sends an SMTP message to a ***Mail Sending Agent*** (***MSA***), which in turn delivers the message to a ***Mail Transfer Agent*** (***MTA***) responsible for transmitting the message to the receiving party. The MSA and MTA frequently reside on the same physical server. The message is transmitted from the sender's MTA to the recipient's MTA, where it is transmitted to a ***Mail Delivery Agent*** (***MDA***) responsible for ensuring the message reaches the recipient's MUA.

The Client-Server Conversation

A client initiates an SMTP conversation over Port 25 with an MSA, such as one managed by the user's ISP. After establishing a TCP connection and receiving the server's banner, the client identifies itself with the HELO command. After receiving an acknowledgment from the server, the client identifies the sender of the message with a MAIL FROM field. Next, the client specifies recipients using the RCPT TO field. Finally, the client provides the message and any attachments in the DATA section, after which the message is sent and the client terminates the connection with the QUIT command. An example SMTP conversation might appear as follows, where the client is notated as “C” and the server as “S”:

```
S:  220 mail.example.com ESMTP Postfix
C:  HELO relay.example.com
S:  250 mail.example.com Hello relay.example.com, pleased to meet you
C:  MAIL FROM:<joe@example.com>
S:  250 <joe@example.com> sender ok
C:  RCPT TO:<alice@othersite.com>
S:  250 <alice@othersite.com> recipient ok
C:  DATA
S:  354 enter mail, end with "." on a line by itself
C:  From: "Joe Smith" <joe@example.com>
C:  To: "Alice" <alice@othersite.com>
C:  Subject: Sample SMTP conversation
C:  This is an example of an SMTP conversation.  Hope you like it.
C:  .
S:  250 Mail accepted for delivery
C:  QUIT
S:  221 mail.example.com closing connection
```

Next, MSA sends this message to an MTA, which then queries the domain name system (DNS) (Section 6.1.2) to resolve the IP address of the MTA of the recipient. For example, given recipient `joe@example.com`, the sender’s MTA would obtain the IP address for the MTA of domain `example.com`. The sender’s MTA then forwards the message to the recipient MTA with a similar conversation as above, and the MTA transfers the message to the MDA.

The SMTP protocol handles sending mail to servers designed to handle queues of messages, but it is not used to deliver mail to clients. Instead, two other protocols are primarily used, the ***Post Office Protocol*** (***POP***) and the ***Internet Message Access Protocol*** (***IMAP***).

POP is the older of these two and was designed to support clients with dial-up Internet connections. As such, a typical POP conversation involves the client connecting to their MDA, downloading any new messages, deleting those messages from the server, and disconnecting.

IMAP is a newer protocol that provides both online and offline operation. In the online mode, a client connects to a mail server and maintains a persistent connection that allows it to download messages as needed. IMAP also allows clients to search for messages on the mail server based on several criteria, prior to actually downloading these messages. Finally, most IMAP sessions by default leave any email messages intact on the mail server rather than removing them on download.

10.2.2 Encryption and Authentication

None of the protocols above for sending and receiving email has any built-in mechanism to guarantee the confidentiality of email messages. Therefore, any party capable of intercepting traffic via IP sniffing (Section 5.3.4) would be able to eavesdrop on any transmitted email messages in his or her subnet. To provide confidentiality, email can be encrypted in one of two ways: at the transport layer or at the application layer.

The most common technique to safeguard the privacy of email is by encrypting the actual transport of messages rather than their contents. Most mail servers support the use of SSL/TLS (Section 7.1.2), protocols that securely encrypt TCP traffic. These protocols are often used at each level of communication—between the client and the local mail server, between the local and destination mail servers, and between the destination mail server and the recipient. Relying solely on transport-layer encryption protects messages against in-flight eavesdropping, but implies a level of trust in the mail servers handling these messages. For example, an employee of an ISP who has access to that ISP's mail server may be able to read the contents of all email messages stored on that server.

Pretty Good Privacy (PGP)

To provide a stronger level of confidentiality, which protects messages from client to client, the actual contents of the email message must be encrypted. There are several approaches that have been proposed for this purpose. One well-known system is ***Pretty Good Privacy*** (***PGP***), which uses public-key cryptography to encrypt and/or digitally sign email messages. When sending a message to an intended recipient using PGP, the sender encrypts the message using the recipient's public key, so that only the recipient can decrypt the message using his corresponding private key.

Verifying the authenticity of a recipient's public key is important for PGP's security, since otherwise an attacker could potentially trick a sender into using the attacker's public key, for which he has a corresponding private key. PGP relies on the notion of a ***web of trust***, contrasting with the hierarchical model employed by certificate services such as SSL. Instead of employing a chain leading to a trusted root certificate, PGP uses a scheme where each public key can be digitally signed by other trusted users, known as ***introducers***, to attest that the public key actually belongs to the party claiming ownership. The basic idea is that after using the system for an extended period of time, each user will retain a collection of trusted keys, and each corresponding trusted party could take the role of an introducer and verify the authenticity of a new public key. (See Figure 10.5.)

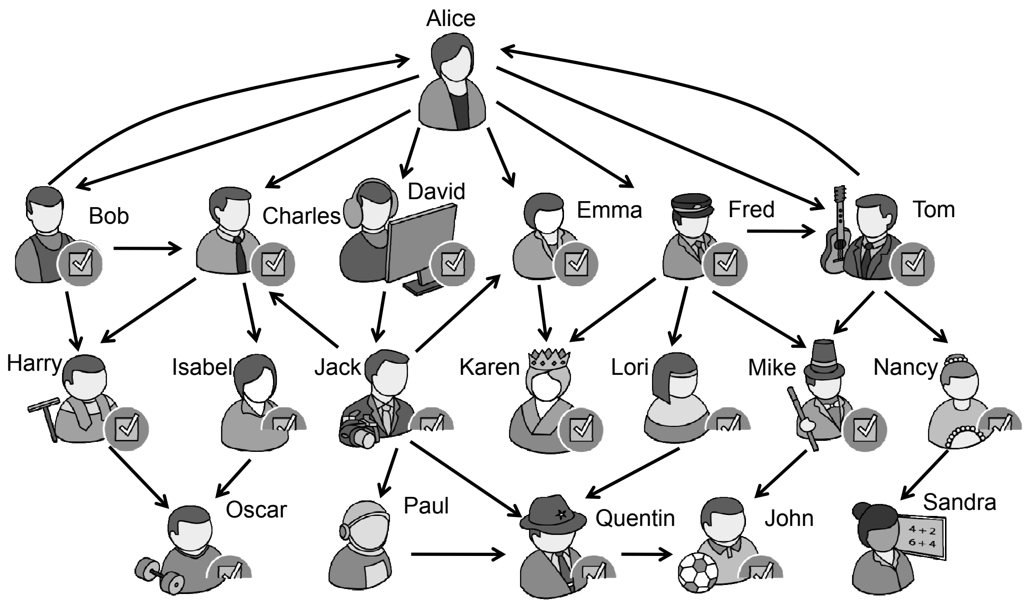


Figure 10.5: A web of trust in PGP. A directed edge from A to B indicates that A signs B's key. A full check mark indicates a key Alice fully trusts and a half check mark indicates a key that Alice partially trusts. People without a check mark or with half check mark have no keys that Alice trusts.

Authentication

The two main approaches currently being used to authenticate the origin of an email message include:

- Authentication of the ***sending user***. This approach allows a recipient mail server to identify the author of an email message. To be effective, however, it requires a widespread deployment of private-public key pairs for mail users. For this reason, it is seldom used in practice.

- Authentication of the ***sending mail transfer agent***. This approach typically identifies the author's organization, but not the individual author. It is simpler to deploy than sending user authentication and has growing adoption.

A complication arises with all types of signed email messages, of course, since even inconsequential modifications while in transit, such as change of encoding, will cause the signature verification to fail. Thus, the body of signed email messages should be formatted in a way that reduces the risk of modifications during transport. This formatting process is called ***canonicalization***.

Sending User Authentication: S/MIME

An email message can be digitally signed to authenticate the sender. For this approach to work, the MUAs of the sender and recipient need to support the cryptographic operations associated with signing and verifying and must agree on the cryptosystem used. The verification of a signed email message relies on the knowledge by the recipient of the public key of the sender. This key can be delivered to the recipient through a secure channel or can be attested by an authority trusted by the recipient.

In the ***S/MIME*** standard for authentication of the sending user, an email is structured according to the ***MIME*** (***Multipurpose Internet Mail Extensions***) standard, which defines the format and encoding of attachments. An S/MIME message has a body consisting of two parts:

- The first part is the message itself, which can consist, in turn. of multiple parts, such as text and attachments.
- The second part is the signature over the first part.

The the structure of an S/MIME message is shown in the schematic example of Figure 10.6(a).

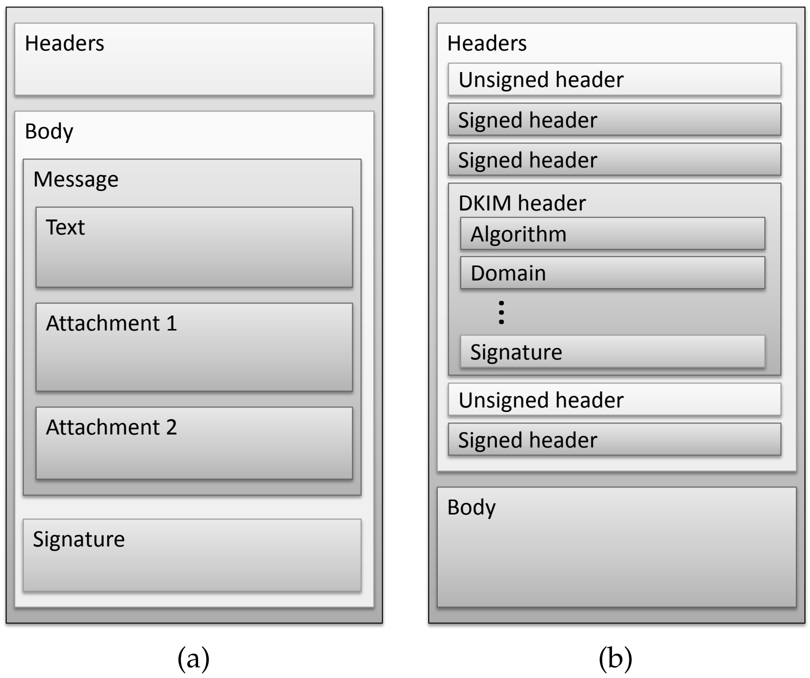


Figure 10.6: Digitally signed email messages: (a) Structure of an S/MIME message, where the signature part refers to the rest of the message body, but not to the headers. (b) Structure of a DKIM message, where the signature in the DKIM header field refers to the message body and selected headers.

Sending MTA Authentication: DKIM

A first approach for authenticating the sending mail transfer agent (MTA) is ***DomainKeys Identified Mail*** (***DKIM***). In DKIM, a ***signing entity***, usually the MTA of the sender, adds a signature to a message to indicate that it originated from the domain of the signing entity. DKIM relies on DNS (Section 6.1.2) for the distribution of the public keys of the signing entities, which are stored in DNS text records. Thus, DKIM is vulnerable to attacks on the DNS infrastructure (Section 6.1.3) unless DNSSEC is deployed.

The DKIM signature covers not only the body of the message but also selected headers. In particular, the FROM field must be signed. The signature is included in a special header field, called DKIM Signature, which is added to the message.

The attributes of a DKIM signature include the following:

- v: version of the DKIM specification
- d: domain of the signing entity
- s: selector of the signing key within the domain
- a: identifier of the cryptographic algorithms used for signing and hashing, for example, rsa-sha256
- c: canonicalization algorithm, the transformation applied to the message to standardize its format (e.g., remove blank lines at the end) before hashing
- h: list of header fields covered by the signature in addition to the body
- bh: hash of the body of the message
- b: signature

An example of a DKIM Signature header field is given in Figure 10.7.

```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed; d=brown.edu; s=cs;
     h=domainkey-signature:mime-version:received:in-reply-to:references
      :date:message-id:subject:from:to:cc:content-type;
     bh=L+J52L7uTfKTel/+2ywqQMH1eiGvl6tsXjDNAySew+8=;
     b=vE2bvcj8GVHGHeECJA4WJ/t1BRbLBvlTQywbZl/HgFSMRfoIVUvH9lyVeMitOaNMeQ
      C29TNP5fJPphaFhHb9tf8EkJBIojRryWRAl5/r5RgT6z5DLWs8fgHe0wUbWEwBQ+sSTs
      A+vbfuLObS1Gwdxtu81HNOfiSLY0u2CM6R31s=
```

Figure 10.7: DKIM Signature header field.

Benefits of MTA Authentication

One of the insecure aspects of email, dating back to its creation when every user on the Internet was trusted, is that, without MTA authentication, the FROM field in an email message can be set to anything the sender likes. Thus, if a sender claims to be a trusted financial institution, there is nothing in the standard protocol to prevent this. The benefit of MTA authentication, then, is that it makes it harder for a sender to falsify a FROM field, since the MTA has to be willing to sign that field as being valid for the senders this MTA is responsible for.

Increasingly, webmail services, such as Gmail, are adopting DKIM to sign both the body and message headers of outgoing content. In addition, many webmail services have begun to reject messages that have not been digitally signed. For example, Gmail now rejects all messages claiming to be from the eBay and PayPal domains unless they have a valid DKIM signature verifying their origin. These steps are effective at eliminating spam (Section 10.2.3) and phishing (Section 7.2.2) attempts claiming to originate from these domains.

Sending MTA Authentication: SPF and SIDF

The ***Sender Policy Framework*** (***SPF***) follows an alternative approach to the authentication of the sending MTA, where cryptography is not employed. The IP addresses of the MTAs authorized to send mail for a domain are stored in a DNS text record for that domain. The receiving MTA checks that the IP of the sending MTA is in the list of authorized IP addresses for the sender's domain, as specified in the MAIL FROM SMTP command. SPF relies on the IP address of the sending MTA. Thus, it is vulnerable to IP source spoofing attacks and DNS cache poisoning attacks. A limitation of SPF is that it does not support mail forwarding. Also, SPF does not protect the integrity of the body of the message.

In comparing SPF with DKIM, we observe that SPF is channel-based and authenticates the sender domain provided in the SMTP envelope, whereas DKIM is object-based and can authenticate the sender domain provided in the From header field. Advantages of SPF over DKIM include faster processing and simpler implementation due to the lack of cryptographic operations at the sending and receiving MTAs. Disadvantages of SPF over DKIM include the lack of support for mail forwarding and for content integrity. Both SPF and DKIM are vulnerable to attacks on the DNS infrastructure.

The ***Sender ID Framework*** (***SIDF***) is similar to SPF. It also verifies the sender's domain specified in the header, such as in the FROM or SENDER fields.

10.2.3 Spam

Since the earliest days of email, advertisers have attempted to capitalize on the ease with which email allows access to millions of potential customers. ***Spam*** email, formally referred to as ***unsolicited bulk email***, is any form of email that is sent to many recipients without prior contact. Spam most often contains advertisements, but can also have more nefarious motives, such as phishing and other attempts to perpetrate fraud. Depending on the country, spam can be of questionable legality, but enforcing laws banning the sending of spam has proven difficult, given the global nature of the problem. Spam is so widespread that it is estimated to account for about 94% of all email sent.

For advertisers, spam is appealing because unlike nonelectronic mail, the majority of the costs associated with sending spam are placed on the recipients, who are forced to store and process the email. For large organizations, this cost is not trivial. At the time of this writing, it is estimated that spam costs businesses around $100 billion per year.

Besides this massive financial burden, spam can be a hassle for the end user, ranging from an inconvenience to an outright threat. Spam is often a vector for scam artists, a means of propagating malware through email, a starting point for phishing attacks, or an attempt at social engineering in the hopes of tricking a recipient to perform some ill-advised action. Because of these factors, a wide range of techniques have been developed to combat spam and prevent it from reaching the end user. In this section, we discuss some of the techniques used by spammers and we explore some prevention measures that can be applied to battle spam.

Harvesting Addresses

There are several techniques by which spammers acquire mailing lists. Some automatically harvest addresses by using specially designed programs that crawl the Web and collect anything that resembles an email address, a process known as ***spidering***. Individuals can often thwart unsophisticated spam harvesters by only posting their email address in a modified form, such as john (dot) smith (at) example (dot) com, which is easily understood by humans but may be difficult to automatically detect.

In addition to automatically searching for email addresses, spammers often buy and sell email lists from other spammers, advertising partners, or criminal networks. For this reason, users are encouraged to give out an email address only to trusted parties, and to review any web site's privacy policy when deciding whether or not to provide an email address to that web site.

Sending Spam

Spammers employ many methods to facilitate sending massive amounts of email. The most common technique involves hiding the origin of email by simply spoofing the FROM field of the message. While this may fool the average recipient, the IP address of the sender's SMTP server is also included in the email header, so any further investigation would reveal this spoofing.

Open Relays and Proxies

If spammers sent mail from an ISP mail server directly, recipients would most likely complain to that ISP, who would in turn shut down the spammer's accounts. Instead, most spammers add a layer of misdirection by sending spam via a third party. An ***open relay*** is an SMTP server which is configured to send email from any recipient, to any destination, in contrast to most ISP mail servers, which only forward email on behalf of their customers. Spammers can use open relays to send their mail without relying on ISP mail servers. However, the dangers of running an open relay are widely recognized, so today very few mail servers allow this behavior.

Another common technique used by spammers relies on ***proxy servers***, that is, servers that act as middlemen in performing connections between pairs of Internet users. For example, when one party sends another party a message via a proxy server, the message appears, to the recipient, to have originated from the proxy rather than from the true source. ***Open proxies*** are servers with this functionality that can be freely used by anyone on the Internet. By sending mail via open proxy servers, spammers can hide the true source of their messages. In order to trace spam back to its source, investigators would need to analyze logs from the proxy server, which could be anywhere in the world and may not cooperate without government intervention. While open mail relays serve few legitimate purposes, open proxies are usually hosted by people wishing to provide users with the ability to browse the Internet anonymously and are not inherently insecure or malicious.

CAPTCHAs

The growing popularity of webmail has provided spammers with a new strategy. Spammers can simply register an account with a free webmail service and use that account to send spam until the webmail provider detects this activity. Many spammers have automated this process by creating programs that register webmail accounts, send as much mail as possible, and repeat the process when the account is cancelled.

To combat automated email account creation tactics, most webmail services require users to solve a ***CAPTCHA*** (***Completely Automated Public Turing test to tell Computers and Humans Apart***). Such a task is anything that is easily solved by a human but is difficult to solve programmatically by a computer. Most CAPTCHAs are image recognition problems, where a distorted image containing a line of text is presented, and the user must interpret the embedded text. (See Figure 10.8.)

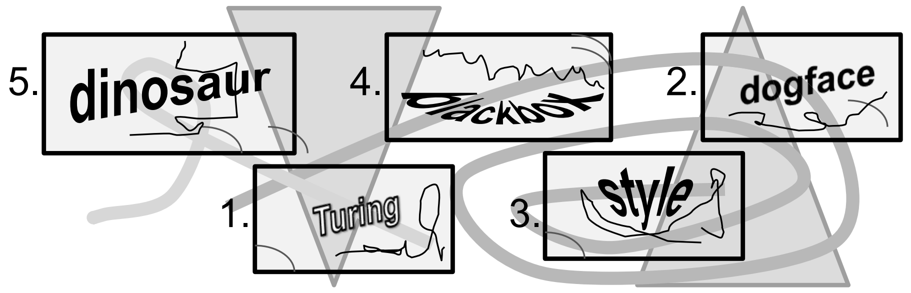


Figure 10.8: A CAPTCHA. Asking a user to type the words they see inside the rectangles, in the specified order, is something that is relatively easy for a human to do compared to a computer.

Unfortunately, some spammers circumvent these CAPTCHAs using web sites that require visitors to solve a CAPTCHA to gain access. Unbeknownst to the visitors, these CAPTCHAs are actually copied from webmail registration pages. The user-provided solutions are then passed to automated spambots in order to register a webmail account for sending spam. In addition, some spammers even employ low-paid workers from developing countries to solve CAPTCHAs for them. In either case, however, the use of CAPTCHAs increases the operational expenses of spammers; hence, these techniques are having a positive effect.

Spam and Malware

Frequently, computers infected with malware are used to send spam, which allows hackers to turn their victims' machines into a means of making money. In fact, it is estimated that over 80% of all spam originates from botnets, which are networks of compromised computers controlled by a single attacker (see Section 4.3.5). Even when botnets are not involved, many viruses turn their hosts into spambots that churn out millions of emails a day. Other viruses turn their hosts into open proxies that spammers use to anonymize their mail. Such spam emails are harder to detect, of course, since they are coming from bots impersonating legitimate users.

The Economics of Spam

Ultimately, the reason spam continues to saturate inboxes with junk mail is because it is profitable for spammers. To analyze the profitability of spam, we must examine a number of factors. The primary cost associated with sending spam is the expense of maintaining email lists, which may be especially significant if lists are obtained by purchasing them from other parties.

Sending email incurs little expense on the sender because nearly all of the operational costs associated with storing large volumes of information are forced on the unwilling recipients.

Other operational expenses for spammers may include acquiring (or renting) and maintaining botnets and mail servers. Finally, the risks associated with sending spam, including criminal prosecution, should be factored into a model analyzing the economics of spam.

Spam is profitable because the total return is generally greater than the sum of these expenses. The ***conversion rate*** refers to the percentage of spam recipients who follow through and perform some desired action that results in the spammer receiving money. This action may be, for example, purchasing a product, signing up for a service, or simply clicking an advertisement, which could generate advertisement revenue for the spammer. The conversion rate is typically extremely small. An experiment conducted by infiltrating a botnet resulted in 28 conversions out of 350 million message, yielding conversion rate of 0.000008%. In general, researchers estimate that the average conversion rate for spam is less than 0.0001%. Nevertheless, despite this narrow turnover rate, the sheer number of recipients allows spammers to recover their expenses and be profitable. (See Figure 10.9.)

Yes, as a matter of fact, I ***am*** a citizen and I ***do*** like the picture you sent.
That penny stock looks like a good investment for our nest egg.
A princess in Nigeria wants to send me money!

Figure 10.9: Dramatizations of the 0.0001% of spam recipients who actually respond to spam emails.

A simple way of modeling the expected profit, P, of a spammer can be described using the formula

$$P = C \cdot N \cdot R - O,$$

where C is the conversion rate, N is the number of recipients, R is the return on each converted email, and O is the total of all operational expenses, including both monetary investments and estimated risk. As a first defense against spam, filtering techniques have been developed to reduce the value of N. In addition, user education programs can help reducing C.

Blacklisting and Greylisting

One of the most popular means of preventing spam from reaching end users is by ***blacklisting*** known and suspected sources of spam and filtering incoming email based on these lists. While maintaining an accurate blacklist would be impossible for any single ISP, there are several centralized resources devoted to aggregating lists of spam sources, which can then be downloaded by mail providers to assist in spam filtering.

Spam blacklists are often published using the domain name system (DNS), in which case they are referred to as ***DNSBLs*** (DNS blacklists). These have been considered controversial, since many DNSBL publishers take a proactive stance against spam and blacklist aggressively, potentially preventing legitimate sources of email from reaching their destinations. Supporters argue that aggressive blacklisting could force ISPs who tolerate spammers to be held accountable for their negligence, while opposers are concerned by the potential impact on free speech over the Internet.

Another spam-filtering technique, known as ***greylisting***, involves the recipient mail server initially rejecting mail from unknown senders. When receiving an email from an unknown sender, the receiving mail server sends a "temporary rejection" message to the sender and logs appropriate information. Since this temporary rejection message is a standardized part of the SMTP protocol, legitimate senders should respond by retransmitting the rejected email after a certain period of time, at which point the receiving mail server will accept the message.

This tactic relies on the fact that spammers are typically trying to send email to millions of recipients, and do not have the resources to handle these temporary rejections and retransmissions. Greylisting is typically very easy to configure and requires no further interaction from an administrator once it is set up. While this is still in accordance with the SMTP protocol, users may desire near-instantaneous mail, which greylisting prevents. Nevertheless, this is a trade-off many administrators are willing to make, especially given how effectively greylisting reduces spam.

Content Filtering

The final antispam mechanism we discuss is perhaps the most complex, ***content filtering***. In this technique, network administrators deploy applications or extensions to mail servers that analyze the text and attachments of each incoming email, determine the likelihood of each email being spam, and perform actions based on this assessment. A naive form of content filtering simply uses lists of blacklisted words and labels a message as spam if it contains any of these words. This sort of scheme may provide basic spam protection, but it usually results in a high number of false positives, where legitimate emails are mislabeled as spam, and false negatives, where spam emails are labeled as legitimate just because they avoid spam keywords, for example, by using disguised words like "V1agr@."

To provide better results, more sophisticated methods of categorizing emails based on their contents have been developed. One of the most effective techniques is known as ***Bayesian filtering***, which relies on a machine learning algorithm to gradually figure out over time how to differentiate spam from legitimate email. In order to achieve this "learning," the filter is first subjected to a training period where it simply records whether or not an email is considered spam based on user responses. The filter maintains a list of all words found in the contents of these emails, and calculates the probabilities that an email containing each word is either spam or legitimate. Once these probabilities have been calibrated over a period of time, the filter can assign a rating to each incoming email that represents its likelihood of being spam. An administrator would then set a threshold, and if an email has a spam rating higher than this threshold, an appropriate action is taken, such as blocking the email entirely or moving it to a quarantine area.

Recent research in spam-filtering has resulted in a number of techniques that seek to utilize user collaboration to categorize and block spam. In this setting, however, care must be taken to ensure that each user's contribution does not violate the privacy of his or her emails. To achieve this goal, systems such as ***ALPACAS*** (***A Large-scale, Privacy-Aware Collaborative Antispam System***) pioneer using a specially designed transformation function that is performed on each examined email to generate a "fingerprint" for that particular message. Ideally, it would be computationally infeasible to determine the contents of a message from its fingerprint, analogous to a one-way hash function. In addition, evasion techniques employed by spammers that subtly alter the contents of each spam message should have no effect on its fingerprint. Systems such as ALPACAS have been shown to be more effective than traditional Bayesian filtering, and may be implemented more widely in the future.

10.3 Payment Systems and Auctions

10.3.1 Credit Cards

Most online sales are completed using credit or debit cards. An online credit card transaction consists of several phases, which involve several parties: the customer, the customer's bank, called ***issuer***, the merchant, the merchant's bank, called ***acquirer***, and the card network (e.g., MasterCard), called the ***card association***.

In the ***authorization*** phase, (see Figure 10.10), the customer provides to the merchant the credit card number along with additional information, such as expiration date and security code. The merchant submits the transaction to the acquirer, which forwards it to the issuer via the card association. The issuer verifies the validity of the card and the availability of funds in the customer's credit line. If the verification succeeds, the issuer decreases the customer's credit line by the purchase amount and sends back to the merchant a transaction authorization via the card association and the acquirer. The authorization phase takes place in real time. Once the merchant receives the purchase authorization, it sends the purchased goods to the customer.

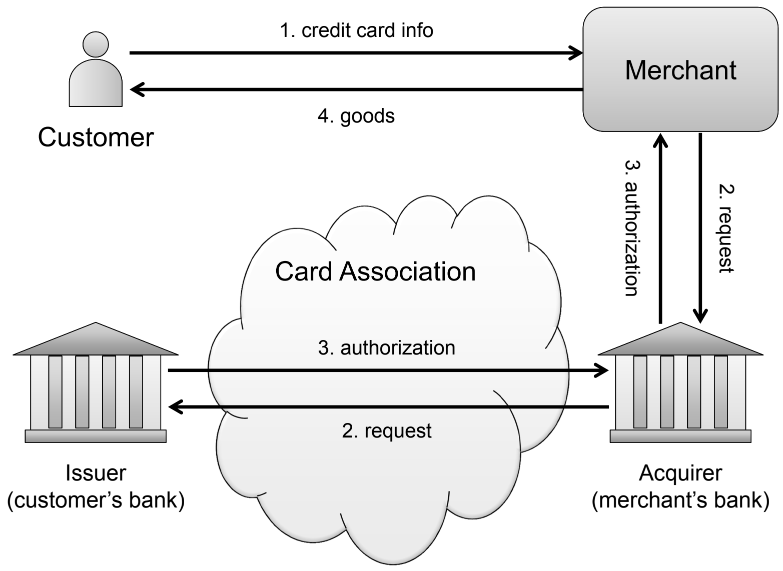


Figure 10.10: The authorization phase in online credit card processing.

Periodically, e.g., at the end of each day, the merchant submits to the acquirer a batch of authorized transactions. The acquirer forwards them to the card network, which handles the ***settlement*** of all transactions. As part of the settlement, the acquirer is credited for the purchase amount and the issuer is debited for the same amount. Once the settlement is completed, funds are transfered from the issuer to the acquirer, the merchant receives the funds and the customer is billed. The settlement takes one to three days from the submission of authorizations to the delivery of funds to the merchant.

Credit Card Fraud and Chargebacks

One of the easiest types of credit fraud comes from the fact that credit cards are first and foremost physical objects that represent something that exists in the electronic world—a line of credit. In an online credit card transaction, the online identification the customer provides is the credit card number and possibly a security identifier, both of which are obtainable given physical access to the card. If the attacker can obtain these numbers, he can make purchases with the victim's credit card. As a result, stolen credit card numbers have become a commonly traded black market item.

There are several protections in place to defend against and mitigate the impact of credit card fraud. United States law limits the liability of cardholders to $50 in the event of fraud, regardless of how much money was spent. This law protects citizens from financial hardship due to fraud. In addition, every customer has the ability to initiate a ***chargeback*** if a fraudulent or otherwise incorrect purchase appears on their billing statement. In the event of a chargeback, the merchant is given the opportunity to dispute the claim, at which point the case would be mediated by both the merchant and customer banks. If the chargeback is undisputed or if the customer's bank wins a dispute, the money for the transaction is refunded to the customer and the merchant must pay a chargeback penalty. Importantly, even if a merchant is not responsible for the fraudulent charges, as in the case of credit card theft, they are obligated to refund the customer. This measure puts strong pressure on merchants to verify the identity of customers before authorizing a purchase, and protects consumers from financial hardship in the event of fraud.

Card Cancellation

Credit card issuers have dedicated phone numbers for the cancellation of a lost or stolen card. Once a card is canceled, all transactions that use the card are denied. Also, attempted transactions are recorded to assist in tracking down abusers. To further protect consumers, banks monitor customer purchasing patterns and apply fraud detection techniques to determine the likelihood that a given purchase is fraudulent. Indicators include consecutive purchases in geographically distant regions and purchase amounts much larger than past averages. In such cases, banks typically place a temporary hold on the account in question until the legitimacy of each questionable transaction can be confirmed by the cardholder.

Cardholder Authentication

Several methods have been devised to provide an additional layer of security on top of the credit card authorization protocol. In the ***3D Secure*** system, implemented by both MasterCard and Visa, the cardholder shares a secret with the issuer and is asked to prove possession of this secret to the issuer when an online purchase is attempted.

In the simplest version of 3D Secure, the customer registers with the issuer to establish a password associated with the card. During an online purchase, the customer is asked to enter the password into a web form that appears in a pop-up window or in an iframe embedded in the merchant's page. This web form is submitted to the issuer, and not to the merchant. The password is used by the issuer as evidence that the legitimate cardholder initiated the transaction.

While aimed at providing an additional layer of fraud prevention, 3D Security may be confusing for the customer. Also, it opens an additional avenue for phishing attacks aimed at capturing the cardholder's password. A further problematic issue is that banks may use 3D Secure as a mechanism to shift liability to the customer in case of fraudulent transactions.

Prepaid Credit Cards

Prepaid credit cards, also known as ***stored-value cards***, are becoming an increasingly popular alternative to traditional credit and debit cards. Unlike credit cards, which allow owners to make charges on credit, or debit cards, which are linked to a banking account, prepaid cards are initialized with a specified balance before being issued. This balance is typically not linked to a bank account, and the card can either be issued to an individual or be used anonymously, depending on the card issuer. Since no credit line or minimum balance is necessary to open an account, prepaid credit cards are commonly used by minors. While prepaid cards may be convenient to use as an alternative to cash, they often provide limited or no fraud protection due to the limited potential impact of fraud—a thief can only spend as much money as resides on a stolen card's balance.

10.3.2 Digital Cash

Digital cash is an electronic currency with the same ***anonymity*** and ***untraceability*** properties of physical cash. Digital cash transactions feature a payer, a payee, and possibly a bank. The basic unit of digital cash is referred to as an ***electronic coin*** or, simply, ***coin***. There are several security goals that a digital cash scheme should meet:

- ***Privacy***. Electronic coins cannot be traced to the payer or payee, mirroring expectations associated with physical cash.
- ***Integrity***. Electronic coins cannot be forged or duplicated, and legitimate transactions are honored.
- ***Accountability***. Transactions cannot be denied at a later date and disputes over transactions can be efficiently settled.

It is easy to ensure that coins can only be produced by valid sources—a simple public-key, digital-signature scheme could be used to verify the authenticity of coins to the merchant. It is difficult to ensure privacy, however, because the bank could match withdrawals with subsequent payments. In order to provide privacy, ***blind-signature schemes*** are often used, which allow a party, in this case the bank, to digitally sign a message without learning the contents of the message itself. In a simple digital-cash scheme, the bank performs a blind signature on the coins withdrawn by the customer. After receiving the coins from the customer, the merchant verifies the digital signature and deposits the coins. During this exchange, the first bank never gains enough information to associate that particular withdrawal with its subsequent deposit.

Preventing ***double spending*** is a more subtle problem. Indeed, it is hard to stop someone from copying electronic coins and spending them in more than one place. In online systems, double spending can be prevented by allowing banks to revoke coins that have been spent, rendering them invalid. For offline systems, one solution relies on identity exposure to prevent double spending. Each withdrawn coin contains encrypted information about the customer's identity, and each deposited coin contains encrypted information about the merchant's identity. With each deposit, a piece of this embedded information is revealed, therefore, a single deposit does not reveal any identifying information. However, subsequent deposits result in a high probability of loss of anonymity.

Several cryptographically secure digital cash schemes have been developed. However, their practical adoption has been rather limited due to lack of sponsorship by governments and financial corporations, which aim at monitoring as much as possible money flows.

Blind Signatures with RSA

The RSA cryptosystem can be used to implement a simple blind signature scheme. Our description below assumes basic mathematical knowledge of modular arithmetic (Section 8.2.1) and the RSA cryptosystem (Section 8.2.2).

Denoting the public modulus with n and the decryption exponent with d, we recall that the RSA signature on a message M is given by

$$\sigma(M) = M^d \bmod n.$$

The customer picks a random coin identifier, x, and a random number, r, relatively prime to n. The pair (x, r) represents a secret coin. Next, using the public modulus, n, and the public encryption exponent, e, the customer computes the value

$$y = r^e x \bmod n$$

and submits it for signing to the bank. Note that the bank cannot retrieve the coin identifier, x, from value y because of the "blinding factor" r^e.

Suppose the bank is willing to sign the value y provided by the customer. Given signature $\sigma(y)$ on y, the customer can derive the signature $\sigma(x)$ on x, as follows:

$$\sigma(x) = \sigma(y) r^{-1} \bmod n,$$

where r^{-1} denotes the multiplicative inverse of r modulo n.

To show that the above formula works, we recall that by the definition of exponents e and d in the RSA cryptosystem, we have

$$ed \bmod \phi(n) = 1. \tag{10.1}$$

Also, we recall that by Euler's theorem, we have

$$a^b \bmod n = a^{b \bmod \phi(n)} \bmod n. \tag{10.2}$$

Using Equations 10.1 and 10.2, we obtain

$$\begin{aligned} \sigma(y) r^{-1} \bmod n &= (r^e x)^d r^{-1} \bmod n = r^{ed-1} x^d \bmod n \\ &= r^{ed-1 \bmod \phi(n)} x^d \bmod n = x^d \bmod n = \sigma(x). \end{aligned}$$

To assure that it is signing a valid coin and not something else, the bank asks the customer to generate k coins and provide cryptographic hashes for each of them. The bank randomly selects a coin and signs it. Also, the bank asks the customer to reveal the remaining $k-1$ coins. The bank then verifies that each such coin hashes to the value provided earlier by the customer. If the verification succeeds, the coin signed by the bank is valid with probability

$$1 - \frac{1}{k}.$$

10.3.3 Online Auctions

Web sites, such as eBay.com, have made ***online auctions*** a viable business model for both individuals selling single items and retail companies with large inventories. Online auctions have many advantages over traditional fixed-value sales and in-person auctions. Like other means of online sale, online auctions expand the customer base to a global market and allow instant and easy exchanges of money and goods. In particular, auctions have the additional advantages of encouraging competition between consumers until the highest mutually agreeable price is determined.

Even so, the anonymous nature of the Internet introduces security concerns to online auctions. First, since any party can register as a merchant, there must be a mechanism to hold merchants accountable for fraud and theft. Online auction sites typically rely on ***reputation systems*** to provide confidence in the legitimacy of merchants. In this case, customers have access to a list of reviews and ratings associated with each merchant, and can rate them with regard to issues of honest portrayal of goods, prompt delivery, and fraudulent behavior. Upon completing a transaction, each customer is asked to provide feedback on the merchant, in order to allow future customers to assess the honesty, integrity, and professionalism of this merchant. Merchants who violate rules are immediately held accountable for these violations via customer feedback, and repeated offenses may result in reduced sales due to low feedback ratings, suspension of account privileges, or potential legal action. Similarly, customers who enter winning bids on items are legally bound to complete the purchase of those items, and are held accountable for this contract. Buyers as well as sellers are rated and held accountable by the reputation system. Care should be taken, however, to prevent buyers and sellers from holding their reputation scores for ransom to obtain extra services or payments.

Another concern for online auctions is ***shill bidding***, which is the practice of a merchant recruiting third parties to fraudulently bid on one of that seller's listed items, with the intent of inflating the current price or perceived desirability of that item. While most auction sites have strict policies banning shill bidding, in reality it is difficult to distinguish shill bids from legitimate bids. Shill bidding detection is far from an exact science, but key indicators may be the use of a newly created account to place bids, frequent bid retractions, accounts that only bid on a limited pool of sellers, and lack of feedback from sellers. Several major auction sites use sophisticated statistical inference techniques to detect shill bidding, but at the time of this writing such methods are kept secret and details of these detection algorithms have not been made public.

10.4 Digital-Rights Management

With all the media that has been digitized, there is a serious concern about how to protect the copyright holders of that content. ***Digital-rights management*** (***DRM***) addresses this concern. DRM refers to the practice of restricting the capabilities users have with respect to digital content. DRM schemes are frequently applied to digital media, such as DVDs and downloaded music, as well as licensed software. (See Figure 10.11.) In this section, we address a number of technological and computer-security issues regarding DRM.

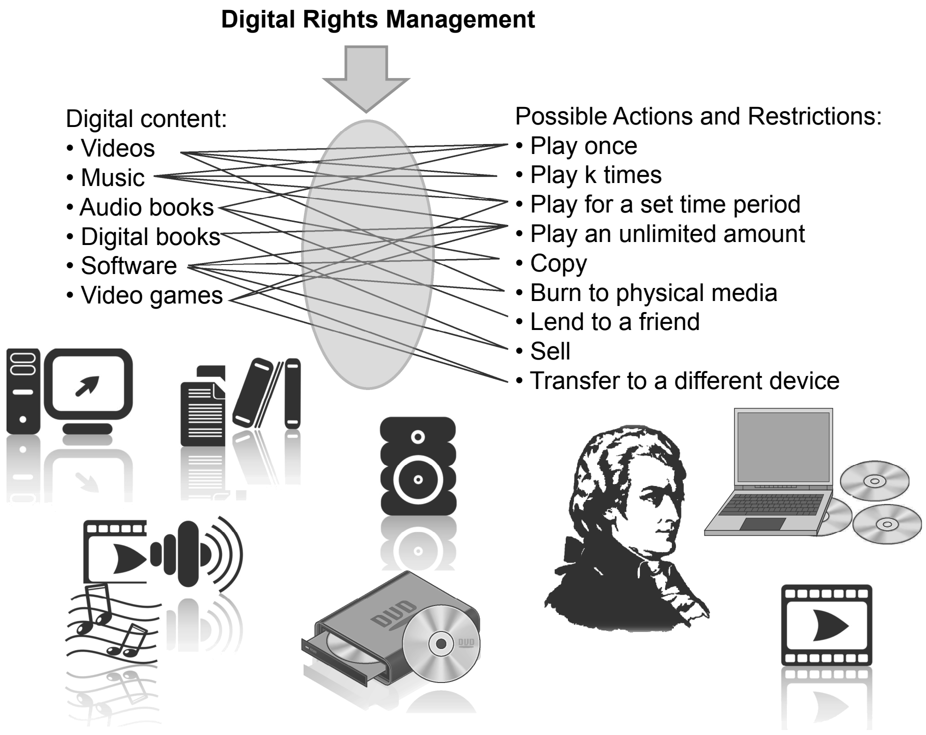


Figure 10.11: Content and possible actions and restrictions that can be applied to that content through digital rights management.

The restrictions that can be imposed through DRM are not without controversy, however, as some people assert that some DRM schemes go beyond the protections provided by copyright law and impinge on the fair use of digital content. So we also discuss some of the legal issues surrounding DRM.

10.4.1 Digital-Media Rights Techniques

A common applications of DRM is protecting digital-media content from unauthorized duplication and from playing on unlicensed devices.

Content Encryption

A simple DRM approach consists of encrypting digital media and storing decryption keys into authorized players. Each media file is typically encrypted with a different key. Thus, the compromise of the key for the specific media object does not affect other media objects. As an additional defense, the encryption key can also be made different for each licensed player, as described below. (See Figure 10.12.)

We consider the scenario where a licensed player downloads a media file from a media server. The player is equipped with a secret ***player key***, P, which is unique to the player and is shared with the server. When the player requests a media file, M, the server generates a random symmetric encryption key F, called ***file key***, and uses it to encrypt the file. Next, the server encrypts the file key with the player key and sends to the player the encrypted file, $C = E_F(M)$ and the encrypted file key, $G = E_P(F)$. To play the media file, the player first decrypts the file key and then uses it to decrypt the media file. That is, the player computes $F = D_P(G)$ and then $M = D_F(C)$.

This simple DRM scheme has the following properties:

- An encrypted media file can be played only by the player that downloaded it. Other players will not be able to decrypt the file key, which is necessary to decrypt the media file. Thus, encrypted media files can be kept in unprotected storage.

- If the file key, F, is obtained by the attacker, it cannot be used to decrypt other media files.

- If the player key, P, is obtained by the attacker, it can decrypt only the media files downloaded by that player.

A first requirement for the security of the system is the strength of the cryptosystem and keys used. A second requirement is that the player should not leak the player key (P), file key (F), or unencrypted media file (M). This requirement is challenging satisfy in a software player, which may be vulnerable to attacks that reverse engineer the code or monitor the program execution to recover the player key.

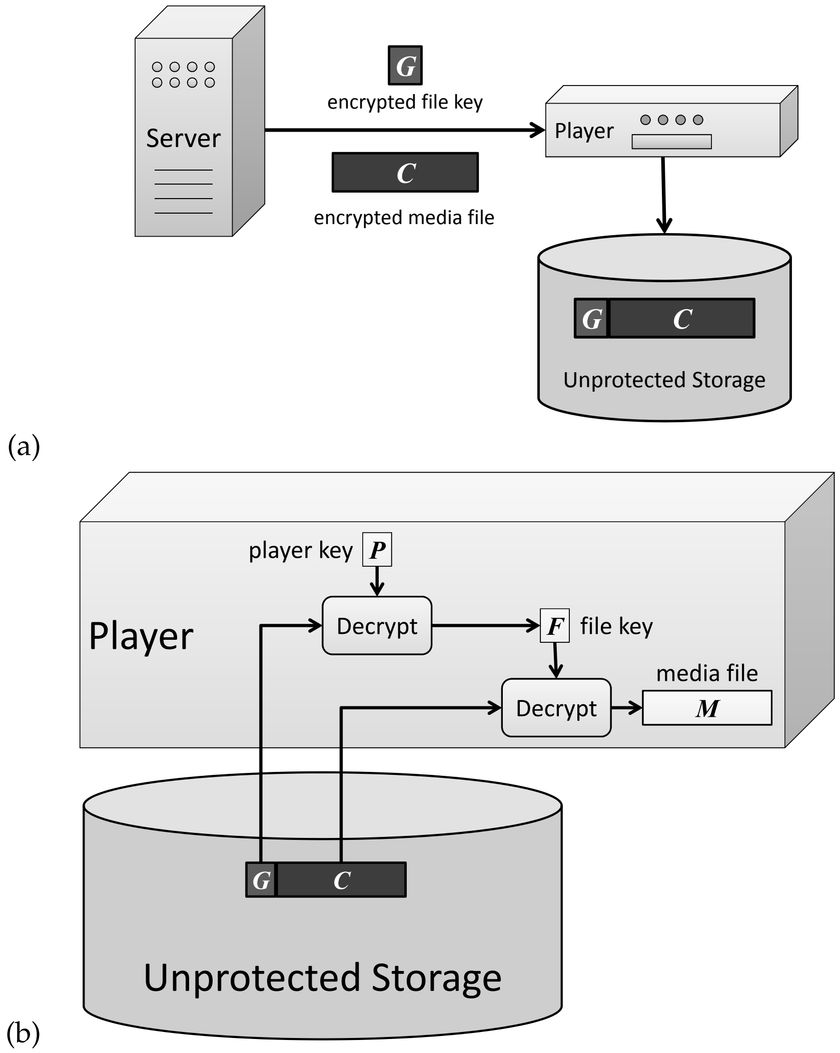


Figure 10.12: A simple DRM scheme for media files: (a) the media server sends to the player the media file encrypted with the file key and the file key encrypted with the player key; (b) the player first decrypts the file key using the player key and then decrypts the media file with the file key.

Key Revocation

Several methods have developed to prevent a compromised player to access any new media content. The ***key tree*** technique views the players as the leaves of a complete binary tree, as shown in Figure 10.13. Each node of this tree is associated with a symmetric encryption key. A player stores all the keys that are on the path from its leaf to the root of the tree. In the example of Figure 10.13, keys K_1, K_2, K_3, K_4, and K_5 are stored by the player associated with the solid-filled leaf, which we refer to as the ***black player***. If there are n players, each player holds $\log n + 1$ keys, where log denotes the logarithm in base 2. The key associated with the root of the tree, which is the only key shared by all players, is used to encrypt file keys.

If a player is compromised, its keys need to be replaced and distributed to the remaining players. In the example of Figure 10.13, the revocation of the black player requires keys K_2, K_3, K_4, and K_5 to be replaced with new keys, denoted K_2', K_3', K_4', and K_5'. The remaining players are subdivided into four groups, denoted G_1, G_2, G_3, and G_4, whose players share keys H_1, H_2, H_3, and H_4, respectively. The rekeying process consists of sending the following four encrypted messages that are broadcast to all players:

$$E_{H_1}(K_2', K_3', K_4', K_5'),\ \ E_{H_2}(K_3', K_4', K_5'),\ \ E_{H_3}(K_4', K_5'),\ \ E_{H_4}(K_5').$$

After rekeying, the black player cannot decrypt any new media files since it does not have the new player key, K_5'. In general, after revoking a player, the remaining players are partitioned into $\log n$ groups, where the players in each group share one key that is not held by the revoked player. Thus, $\log n$ news keys, replacing those stored at the ancestors of the revoked player, can be distributed using $\log n$ broadcast messages sent to all players.

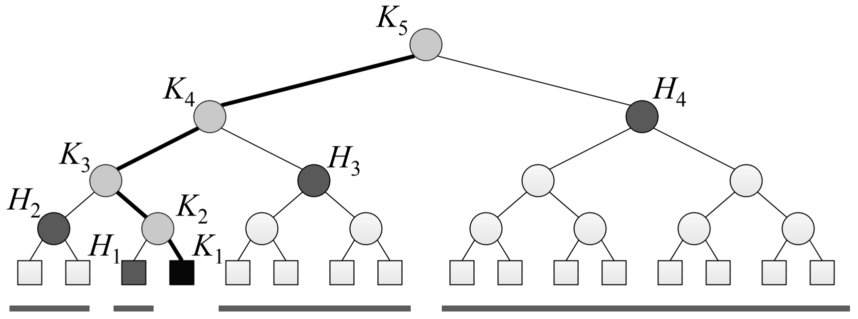


Figure 10.13: Key tree for player revocation. The black player, which is associated with the solid-filled leaf, stores keys K_1, K_2, K_3, K_4, and K_5. To revoke the black player, keys H_1, H_2, H_3, and H_4 are used to encrypt replacements for keys K_2, K_3, K_4, and K_5.

10.4.2 Digital-Media Rights Practice

Until recently, implementations of DRM technology have been mostly unsuccessful. In this section, we review DRM methods used in practice for CDs, DVDs, and downloadable media.

Compact Discs

In 2002, several DRM schemes for audio CDs started being adopted with the goal of preventing the copy of CD contents onto a hard drive or other external media. Compatibility problems often resulted in customers being unable to play their legally purchased music on some devices, and many schemes were eventually reverse engineered and rendered ineffective.

In 2005, Sony BMG generated major controversy by introducing a DRM technology on audio CDs. In the default configuration of Windows XP, inserting a CD causes the software on it to be automatically executed to facilitate software installation by nonexpert users. While software installers typically prompt the user to explicitly launch the installation process, the DRM software installed itself silently. Also, the DRM software behaved similarly to a rootkit (Section 4.3.3), hiding its files and processes. Researchers found that the DRM software included a security vulnerability, unknowingly turning its users into potential targets for exploitation. In response to a major wave of criticism from consumers, and eventually several lawsuits, Sony issued a patch to remove the rootkit and stopped using the DRM technology. Due to the controversy generated by this incident and weaknesses and costs associated with implementing DRM, no major music publishers are currently producing DRM protected CDs.

Digital Video Discs

In contrast to CDs, for which DRM technology is not standardized, nearly all commercially produced DVDs feature a DRM scheme known as the ***Content Scramble System*** (***CSS***). CSS was designed to meet several security goals. First, only licensed DVD players contain the player key necessary to decrypt CSS encrypted disks, which allows for strict regulation. Second, communications between the player and host are encrypted to prevent eavesdropping data in transmission. While the CSS DRM technology was meant to be kept secret, CSS was reverse engineered, published, and broken—yet another confirmation of the failure of security by obscurity. (See Section 1.1.4.) An additional limitation of CSS was due to the fact that the United States enforced strict regulations regarding the export of cryptography at the time of CSS's design. These regulations limited the length of cryptographic keys to at most 40 bits, even if it had already been shown that this key length is insufficient to prevent brute-force decryption attacks (Section 1.3.3).

Since the breaking of CSS, several other DRM schemes have been adopted for various video formats. Blu-ray relies on a sound DRM scheme known as the ***Advanced Access Content System*** (***AACS***), which has a publicly available specification. AACS is based on the strong AES block cipher. Also, it stores multiple keys into each player and incorporates a sophisticated mechanism for revoking player keys that extends the one given Section 10.4.1. Another innovative Blu-ray solution is known as ***BD+***, a technology that essentially embeds a small virtual machine in authorized players and treats Blu-ray content as executable programs that are verified and executed by the player.

Downloadable Media

Apple's iTunes music player allows users to download individual songs or albums through the iTunes store. Songs downloaded in this way can be encoded using ***FairPlay***, a DRM technology that encrypts each track so that only the user who downloaded the file can listen to it.

Several techniques were developed to circumvent FairPlay, to which Apple responded by adjusting FairPlay to render the attack useless. In 2009, Apple announced that it had finally reached an agreement with major record labels to remove DRM restrictions from the iTunes music store. This decision marked a major turning point in policy regarding the distribution of digital music.

The public seems to be somewhat more tolerant of DRM restrictions on digital video. For example, at the time of this writing, Apple has a DRM mechanism that can place a time restriction on movies that downloaded through iTunes, and Netflix uses a subscription model to restrict digital video downloads and viewing to customers with up-to-date subscriptions.

A more recent development, brought on with the advent of handheld document readers, like the Amazon Kindle, Apple iPad, Barnes and Noble Nook, and Sony e-Reader, is the concept of an electronic book, or ***ebook***. DRM technologies have been developed for ebooks in a way similar to those for downloadable music and video. In some cases, reading rights can be modified even after an ebook has been purchased. For example, in an ironic twist, ebook versions of George Orwell's *1984* and *Animal Farm* were remotely removed from the Kindles of some users in 2009 after they had purchased ebook versions of these novels, which warns of the risks of intrusive centralized power.

10.4.3 Software Licensing Schemes

Proprietary software has employed various licensing schemes for decades. ***Software licensing*** is important for software vendors because it provides a means of protecting products from unauthorized use or duplication. Older licensing schemes, which existed before easy access to the Internet, typically require the vendor to provide a registration key or serial number to each customer. The application would offer limited or no functionality until the user was provided with this key. Without communication with the Internet, this simple mechanism does little to prevent piracy, since the same key could be used on any number of copies of the product.

Since offline licensing schemes have no access to the Internet, all of the logic required to verify a registration key must be built into the software itself. It would be ineffective to implement a scheme that simply stores a list of valid keys within the compiled application—such a strategy is easy to defeat if an attacker can successfully reverse engineer the binary code of the application.

Windows Product Activation

Instead of storing actual keys in the data of the program, most licensing schemes dynamically generate keys based on user input or the properties of the machine on which the software is being installed. Microsoft employs these techniques in their product registration process, which they refer to as ***activation***. Since XP, Windows installations will cease to function normally once a specified period of time has passed unless they are activated. The user is provided with a unique 25-character product key on purchase. When the user agrees to perform the activation process, a 72-bit ***product ID*** is derived from the product key using a secret encryption method. Also, a 64-bit ***hardware hash*** is computed from the hardware components of the machine, including the processor type and serial number, the amount of memory, the hard drive device name and serial number, and the MAC address. The product ID and the hardware hash are then stored in the registry and sent to Microsoft.

When Microsoft receives a product ID and hardware hash, it checks that the product ID has been issued by Microsoft and is not forged or pirated. If the product ID is valid, Microsoft issues a digitally signed release code that is stored on the machine. On booting, Windows checks that this release code exists and —if not, the user is informed that they must activate their product or it will stop working. On booting, Windows also checks that the hardware hash created during activation matches the current hardware profile of the system, to prevent a user from activating

Windows on more than one machine. To give the user some flexibility in modifying or repairing their machine's hardware, this check is done using a simple voting scheme. The product activation software tallies a vote for each current device that matches the stored hardware profile. On Windows XP, if seven positive votes are tallied, the confirmation process succeeds and the user may continue using the system. If a user modifies a system in such a way that this verification fails, he must request a new release code from Microsoft directly.

Windows activation is effective because it is integrated into the operating system itself. As such, it is difficult to reverse engineer since the very environment in which any dynamic (performed while the target is running) reverse engineering process might be performed would prevent such analysis. While it may be possible to statically reverse engineer relevant libraries, this would be a complex task given the complexity and size of the codebase. Still, if a similar scheme were integrated into ordinary software, it might be more easily defeated.

A Sample Software Licensing Scheme

Consider the following software licensing scheme, which is similar to several schemes used in practice:

1. When the user purchases a license to the software, the manufacturer generates a random ***license key***, stores it in a registration database, and gives it to the user.
2. The software installer asks the user to provide the license key, which is stored on the machine. Also, it generates a ***machine ID***, which is a cryptographic hash of a string that describes the main hardware components of the machine.
3. The machine ID and the license key are sent by the installer to the software manufacturer, which verifies that the license key is in the registration database and has not been previously associated with another machine.
4. If the verification succeeds, the manufacturer associates the machine ID with the license key in the registration database. Also, it generates a ***registration certificate***, which is a digital signature on the pair (license key, machine ID). The registration certificate is sent to the installer and stored on the machine.
5. Each time the application is launched, it retrieves the license key and recomputes the machine ID by examining the currently installed hardware components. Next, the application verifies the license key and machine ID using the registration certificate and the manufacturer's public key. If the verification fails, the application terminates.

The above scheme defends against several attacks, including forging license keys or registration certificates, installing the software on more than one machine, and reselling the software. Nevertheless, even relatively strong schemes such as this often have one fatal flaw. If an attacker can alter the machine code of the software in question, he may be able to change the program's behavior to skip the licensing process completely. Imagine altering a single conditional statement in the assembly code of the program (perhaps corresponding to "if registration succeeds, continue execution") to an unconditional jump that always results in continuing execution.

Altering a compiled program to bypass protection schemes is commonly known as ***patching***. Situations such as these provide the motivation for ***binary protection schemes*** that include techniques that make it more difficult to deconstruct or reverse engineer an application, such as compression, encryption, polymorphism, and other methods of code obfuscation. Binary protection schemes are similar to the virus concealment schemes discussed in Sections 4.2.3 and 4.2.4. Strong binary protection schemes make it difficult to patch the binary version of a program, making DRM circumvention difficult.

10.4.4 Legal Issues

The widespread adoption of the Internet has created a convenient avenue for piracy of both software and media content. Both legislators and copyright holders have encountered difficulties in creating and enforcing laws to protect artistic and intellectual property in the international arena of the Internet. At the same time, groups such as the ***Recording Industry of America*** (RIAA) have generated controversy by aggressively prosecuting individuals participating in the illegal distribution of music via online file-sharing. At the time of this writing, there are still a number of legal gray areas, such as aggregation web sites that provide access to illegal content hosted by third parties, which raise questions of responsibility.

DRM itself has been the subject of several legal decisions as well. Most notably, the ***Digital Millennium Copyright Act*** (***DMCA***) was passed in 1998, which dictates that reverse engineering and circumvention of a technology designed to restrict access to a work protected under copyright law is illegal if done with the intent of violating that copyright. However, DMCA provides several exemptions from its clauses against reverse engineering and circumvention for educational and research purposes. Overall, DMCA gives copyright holders significant power to protect their content and enforce that protection with the support of the legal system.

10.5 Social Networking

Social networking refers to the use of online communities designed to facilitate contact between groups of people and individuals with general interests or a wide variety of special interests, ranging from dating to job searches to photography.

10.5.1 Social Networks as Attack Vectors

The great benefit of social networking sites, such as Flickr, Facebook, MySpace, LinkedIn, and Twitter, is that they promote a great amount of communication between people identified as "friends." Unfortunately, these increased levels of communication and trust can also act as attack vectors. Indeed, the risks come from several different directions.

First, these web sites typically provide many channels of communication between users, including the ability to be contacted by strangers, who might actually be engaged in information-gathering attacks. The risks of such contacts can be serious, since compromising a user's social networking account may yield access to private information that could be used to facilitate identity theft, fraud, or harassment. This risk is further increased as studies show that as much as 15% of social networking users will reciprocate a friend request from a stranger. Thus, even if personal information is restricted to friends-of-friends, there is a chance that information could still be open to attack if a friend reciprocates a random friend request.

Another attack risk for social networking web sites comes from the fact that they are highly interactive, dynamic web applications. For instance, several social networking web sites allow third parties to write applications that run inside the security domain of the site. Even if the software base for the web site is secure, these third-party applications are potential attack vectors. Thus, administrators for social networking web sites should have stringent vetting processes in place for third-party applications.

In addition, because they support various kinds of interactive user communication, social networking web sites are potential vectors for cross-site scripting attacks (Section 7.2.6). Such attacks can leverage code executed in a victim's browser to propagate XSS worms, links to malware, or spam advertisements. Moreover, because users place some degree of trust in their social networking peers, attackers can exploit this trust to distribute malware or spam via compromised accounts. Such a compromise may be a result of a phishing attack, data theft due to malware on a victim's machine, or even a breach of the social networking service itself.

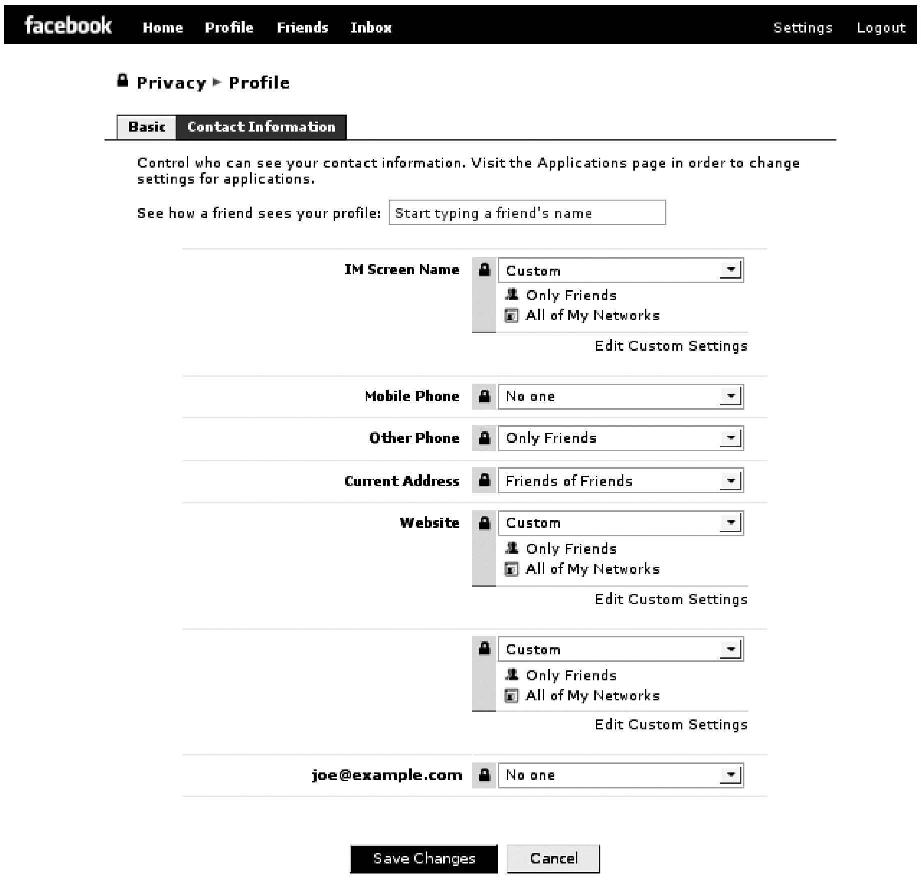
10.5.2 Privacy

With the growing popularity of social networking web sites, people are more frequently making personal information public and visible to at least some portion of the Internet. When taken in aggregation, social networking sites can often allow untrusted parties to build alarmingly complete profiles on a person. For example, it is not uncommon for employers to search for personal information on social networking sites to gather additional data on prospective job candidates. This undesired disclosure of personal information can be dangerous, in fact, because young children are increasingly using social networking sites. Intimate personal details and a mechanism for initiating contact with strangers can provide an easy means of access for predators and fraudsters.

Because of these risks, social networking sites must take three important steps to protect the privacy of their users. First, users must be given complete control over what personal information is available to what parties. These options must be easily accessible to users, and extremely simple to configure. Figure 10.14 depicts Facebook's privacy settings page at the time of this writing, as an example of a system that has undergone repeated changes to make configuration easier for users. Accordingly, users have some degree of responsibility in carefully considering the extent to which their personal information is disclosed.

Second, privacy settings must be assigned restrictive default values to protect users who are unwilling or unable to configure their own privacy preferences. For example, sites sharing personal details should default to only making those details available to parties with which the user has explicitly initiated contact. Such restrictions are especially important for protecting young children who may not be aware of the dangers of disclosing too much personal information to the public Internet or may be unable to properly configure their own settings.

Finally, social networking sites have an obligation to clearly dictate policies regarding sharing of user information. Users should be aware of how their personal information can be accessed and used by third parties. For example, the social networking site Quechep faced harsh criticism for automatically sending invitations to the entire email address book of each user, without asking permission. Other less reputable sites go so far as to sell email addresses and personal information to spammers. Explicit privacy policies allow users to hold social networking sites accountable for these actions.


facebook Home Profile Friends Inbox Settings Logout

Privacy ▸ Profile

Basic | Contact Information

Control who can see your contact information. Visit the Applications page in order to change settings for applications.

See how a friend sees your profile: Start typing a friend's name

IM Screen Name	Custom Only Friends All of My Networks Edit Custom Settings
Mobile Phone	No one
Other Phone	Only Friends
Current Address	Friends of Friends
Website	Custom Only Friends All of My Networks Edit Custom Settings
	Custom Only Friends All of My Networks Edit Custom Settings
joe@example.com	No one

Save Changes | Cancel

Figure 10.14: Facebook allows its users to customize the degree to which personal information is shared with other users.

Privacy Risks from Friends Lists

There is an old saying, which has a modern interpretation in the context of social networking:

> "Show me a man's friends, and I will show you the man."

Interestingly, various studies have shown that the mix of one's friends on a social networking web site can contain information that may make it possible to predict, with some degree of accuracy, information about that person, including religion, race, gender, age, and sexual orientation. Thus, even just the mix of one's friends can have privacy implications.

In addition, studies have also shown that it is possible to correlate users between different social networking web sites just by matching up friends lists. Therefore, one should be aware of the risks of having similar sets of friends between a site that allows for seemingly anonymous pseudonyms for usernames and a site where one uses a real name.

10.6 Voting Systems

Electronic voting can be conceptualized as a multiparty computation where each party contributes his or her vote and the result can be totaled from each submitted vote.

10.6.1 Security Goals

There are several security goals for a computational voting system

- ***Accuracy***. The reported results should accurately reflect voter intent.
- ***Availability***. The means to vote should be available to all voters for the entire term of the announced times for voting.
- ***Secrecy***. No party can prove a particular vote was associated with a single individual after the act of voting has taken place, including the voting party.
- ***Verifiability***. Each voting party can confirm that his or her vote was tallied properly, that the reported totals are accurate, and that only authorized voters had their votes counted.
- ***Usability***. The system should be understandable to the average voter. Also, casting voting, tallying votes, and verifying votes should be easy to accomplish.

Of all these requirements, secrecy is most important for preventing ***voter coercion***, where a voter is pressured or rewarded by another party to vote against his or her will. Such influence is reduced by secrecy, since the voter can no longer prove to the third party whether or not he or she voted in a certain way.

Verifiability, on the other hand, helps to prevent ***voter fraud***, where fictitious voters cast votes that are counted in the reported results or actual votes are not counted. If each voter can verify the results, it becomes harder to carry out voter fraud.

Intuitively, verifiability and secrecy seem to be mutually exclusive. How can a party verify that his or her vote was counted properly and still not be able to prove to an outside party what that vote was? Modern voting schemes attempt to address these security goals, while maintaining good usability. We will discuss a recently proposed ***verifiable voting*** scheme designed to satisfy these security requirements, and provide a comparison to the currently implemented election protocol in the United States. See Section 2.5.2 for a discussion on voting machines.

10.6.2 ThreeBallot

ThreeBallot is a computational voting scheme, designed by Ron Rivest, that can be implemented on paper without the use of cryptography. It derives its security from the use of randomization.

Casting Votes

The idea behind ThreeBallot is simple to state, but perhaps a bit nonintuitive. A voter is given three ballots, each with a unique identifier. Each candidate has a single voting bubble on each ballot. The voter is instructed to cast exactly two votes for their preferred candidate and exactly one vote for the remaining candidates. That is, to vote ***for*** a candidate, the voter fills in the bubbles for that candidate on any two of the three ballots. Instead, to vote ***against*** a candidate, the voter fills in one bubble for that candidate on any one of the three ballots. An example of the ballots for a vote in an election with three candidates is shown in Figure 10.15. There are several possible valid configurations for the three ballots. In particular, one of the ballots could be blank. For example, the voter could mark all candidates on the first ballot, mark only the preferred candidate on the second ballot, and leave the third ballot blank.

Given the voter's three ballots, a trusted party must verify that the votes are valid, that is, no candidate is marked on all three ballots or unmarked on all three ballots and only one candidate is marked on two ballots. For example, this trusted party could be a simple ballot checking machine that can be inspected at any time to assure correct operation. After they pass verification, all three ballots are submitted anonymously. Also, the voter is given as a receipt a copy of one of the ballots, which is secretly chosen by the user. This receipt will be used in the vote verification phase.

Vote Tallying and Verification

When all of the ballots have been collected, under the ThreeBallot system, the ballots are posted publicly, the totals are tallied, and the winners and their respective vote tallies are announced. Note that determining voter intent from these totals is straightforward—if a candidate were to receive v votes in an ordinary election, then under this system they would receive $v + n$ votes, where n is the number of voters. This is because v voters cast $2v$ votes for this candidate (as their preferred candidate) and $n - v$ voters cast $n - v$ votes against this candidate, for a total of $2v + n - v = v + n$ votes.

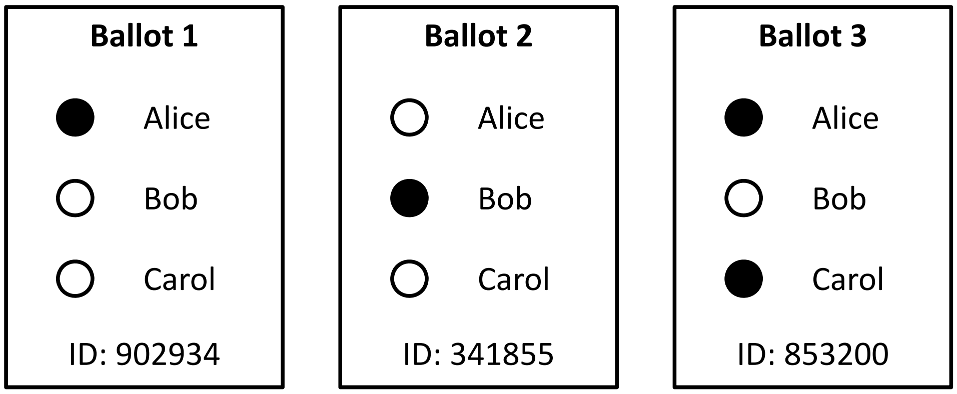


Figure 10.15: In this ThreeBallot election, the voter votes for Alice by marking two of the three ballots at random for Alice, and votes against Bob and Carol by marking only one ballot at random for each of these candidates.

Analysis

The receipt allows a voter to verify that one of her ballots is included in the tally. Because any attempt to alter a ballot has a 1/3 chance of being detected, the probability of successfully perpetrating large-scale vote fraud is extremely low provided enough voters verify their receipt. Namely, assume that m ballots have been modified and that a fraction f of voters ($0 \leq f \leq 1$) verify their receipt. The probability that the tampering goes undetected is

$$\left(1 - \frac{f}{3}\right)^m .$$

For example, if $m = 64$ and $f = 50\%$, that is, 64 ballots are tampered with and half of the voters check their receipt, the probability that tampering goes undetected is less than 0.001%. Thus ThreeBallot provides ***verifiability*** with high probability.

Regarding ***secrecy***, the marks on the receipt do not imply any specific vote. Thus, obtaining receipts is of limited use for an attacker. Instead, in order to buy or coerce a voter, an attacker can ask the voter to place the marks in the ballots according to specific patterns selected by the attacker. The attacker will then confirm the vote by looking for the three patterns in the posted ballots. This attack is effective only if the number of candidates is large enough so that the probability that two ballots have the same marks is very small, which implies that the marks patterns on the ballots essentially identify the voter. Thus, in order to provide secrecy, the number of candidates must be limited depending on the number of voters. For example, for 10,000 voters, there should be at most six candidates. This privacy requirement for ThreeBallot is called the ***short ballot assumption***.

Comparison With Traditional Voting

ThreeBallot provides guarantees that currently implemented voting schemes do not. Traditional elections, such as the presidential election in the United States, lack transparency. Secrecy is provided, since no receipt is given to a voter that would allow him or her to prove which candidate they selected. On the other hand, there is absolutely no verifiability from the perspective of the voter. Auditing an election by recounting ballots is cumbersome and time consuming, and thus done only in exceptional circumstances. Also, the audit verifies only the overall tally but does not give any guarantees about the integrity of the ballots. Indeed, in a traditional election, the election authority is assumed to be a trusted party. This trust is supported by knowledge that the election authority is carefully scrutinized and audited by external parties, but still the average voter has few assurances that his or her vote was counted properly.

The only dimension where traditional voting may be superior to Three-Ballot is usability. The single-ballot system is straightforward and widely understood. Also, it does not require any ballot-checking machine for casting votes. On the other hands, should ThreeBallot become adopted, the learning effort for voters would be rather modest and the cost of deploying and testing ballot-checking machine would be low.

A comparison of ThreeBallot with the traditional US election scheme is shown in Table 10.1.

	US Election	**ThreeBallot**
Secrecy	Yes	Yes
Individual Verifiability	No	With 33% probability
Overall Verifiability	Through auditing	With high probability
Usability	High	Medium

Table 10.1: Comparison of the traditional U.S. election scheme with Three-Ballot. Individual verifiability denotes whether a voter can verify that their individual vote was properly included. Overall verifiability is the ability to ensure that the election authority is tallying votes fairly.

10.7 Exercises

For help with exercises, please visit **securitybook.net**.

Reinforcement

R-10.1 Describe the SQL query that would select from the `Presidents` table all those people whose age at their death was over 70. In addition, describe the SQL command that would delete from the `Presidents` table all those people whose age at their death was over 70.

R-10.2 Explain how the two-phase commit protocol helps to achieve database integrity and availability. Does it also help with confidentiality and privacy? Why or why not?

R-10.3 Suppose the following sequence of SQL commands is executed (and in the following order):
First, By Bob:
```
GRANT SELECT ON employees TO Alice WITH GRANT OPTION;
GRANT SELECT ON customers TO Alice WITH GRANT OPTION;
GRANT SELECT ON accounts TO Alice WITH GRANT OPTION;
```
Then, by Alice:
```
GRANT SELECT ON employees TO Charles WITH GRANT OPTION;
GRANT SELECT ON customers TO Charles WITH GRANT OPTION;
```
Then, by Charles:
```
GRANT SELECT ON employees TO Diane WITH GRANT OPTION;
GRANT SELECT ON customers TO Diane WITH GRANT OPTION;
```
And, then by Bob:
```
REVOKE SELECT ON employees FROM Alice;
```
What access rights do Alice, Charles, and Diane now have at this point?

R-10.4 What is the policy that Alice is using to determine which keys she fully trusts, partially trusts, and doesn't trust in her web of trust, illustrated in Figure 10.5?

R-10.5 What is the solution to the CAPTCHA in Figure 10.8?

R-10.6 Describe all the computer vision problems that would have to be solved in order for a computer to be able to figure out the CAPTCHA in Figure 10.8?

R-10.7 What are the comparative benefits of blacklisting and greylisting of emails?

R-10.8 For each of the following security properties, state whether they are provided by S/MIME and why: (1) Confidentiality, that is, only the recipient of the message can read it. (2) Integrity, that is, changes to the message are detected by the recipient. (3) Sender identification, that is, the recipient is assured of the identity of the user who sent the message.

R-10.9 For each of the following security properties, state whether they are provided by DKIM and why: (1) Confidentiality, that is, only the recipient of the message can read it. (2) Integrity, that is, changes to the message are detected by the recipient. (3) Sender identification, that is, the recipient is assured of the identity of the user who sent the message.

R-10.10 A spammer named Richard has bribed an ISP official $1,000 to let him send out as many spam emails as he wants and he has no other costs. The conversion rate for his spam is the usual 0.001% and he gets $10 for each converted response. What is Richard's expected profit or loss if he sends out 1,000 emails, 100,000 emails, 1,000,000 emails, or 100,000,000 emails?

R-10.11 In the previous exercise, how many emails does Richard need to send in order to be at an expected break-even point, that is, the point where his expected profit is zero?

R-10.12 Describe the main differences between S/MIME and DKIM.

R-10.13 What should you do if you notice a charge on your credit card statement that you are sure you didn't make? Also, what are the actions that happen behind the scenes after you take this action?

R-10.14 What mechanism discourages someone from double-spending their digital cash? Do you think this is an effective deterrent? Why or why not?

R-10.15 What is shill bidding and why should an online auction company care about stopping it? After all, doesn't shill bidding increase the profits for the online auction company?

R-10.16 Describe five reasonable restrictions that a movie company would want to apply to people who rent their films from an online download service.

R-10.17 Name three security risks that are possible in social networking web sites.

R-10.18 Some social networking web sites provide mechanisms for users to determine the GPS coordinates of where their friends are located at any given moment. Describe some security and privacy risks that this technology presents.

R-10.19 What are the key security properties that any computer voting scheme should have?

R-10.20 In the ThreeBallot voting system, if there are 23 candidates running for the same office, how many bubbles does someone have to fill in to correctly vote for their preferred candidate?

Creativity

C-10.1 Suppose that Bob is maintaining a server to store Alice's database and answer SQL queries for this database via the Internet. Alice wants to achieve confidentiality for her database (including confidentiality from Bob himself); hence, she wants to encrypt every cell in her database tables. Describe how she can do this so that Bob can still answer SQL queries to find every record that matches a certain value, like `Inaugural_Age=46.2`, except now the `46.2` will be some encrypted value. Specify in your answer the cryptosystem Alice should use and why, including why the Elgamal cryptosystem would not work for this purpose.

C-10.2 Consider the ***outsourced database*** problem of the previous exercise, but now suppose that there is an attribute, `Age`, in Alice's table for which she would like to do ***range queries***, to select people whose age falls in one of the standard decades, that is, teens, twenties, thirties, etc. Explain how Alice can encrypt all of her values to achieve confidentiality and still allow these types of range queries.

C-10.3 Alice has a table of famous 19th-century people and their exact ages at death, for which she wants to anonymize using generalization, dividing ages into ranges, such as "46.35–48.08," so that each age range has at least 40, but no more than 80, people in it. Describe an efficient algorithm for Alice to perform this generalization, assuming there are no more than 40 people in Alice's table with the same exact age at death.

C-10.4 Describe how an email reading program (email user agent) should handle messages signed with the S/MIME standard. Which notifications should be given to the user? Recall that in the S/MIME email authentication standard, the signature does not protect the headers of the message.

C-10.5 Explain why a DNS cache poisoning attack can compromise DKIM but not S/MIME. Describe how DKIM could be modified to defend against DNS based attacks.

C-10.6 A spammer named Richard pays people in Elbonia \$0.01 for each CAPTCHA they solve, which he can then use to create an email account that can send out 10,000 spam emails before it is shut down. The conversion rate for his spam is the standard 0.001% and he has no other expenses other than the money he pays his employees in Elbonia. What is the formula for Richard to determine his expected profit in terms of N, the number of recipients, and R, the dollar return on each converted responder?

C-10.7 Suppose Alice has a policy that she trusts the key of anyone provided their key is signed by her, signed by someone whose key she has signed, signed by someone whose key is signed by someone whose key she has signed, etc. Draw a diagram for a web of trust having at least 10 people such that Alice trusts everyone but she has signed only one key. Likewise, Draw a diagram for a web of trust having at least 10 people such that Alice trusts no one (other than herself).

C-10.8 Describe an alternative CAPTCHA system, other than twisting words into strange shapes, that would be easy for a computer to generate but hard for a computer to solve.

C-10.9 Alice has a ***whitelist*** solution to her spam problem: she only accepts emails from people who are in her address book. All other emails are rejected. Is this an effective way to block spam? Why or why not?

C-10.10 Describe a rule change that would allow sellers and buyers in an online auction to still provide feedback on their experience but would prevent them from holding their feedback for ransom in response to first getting a positive feedback from the other party.

C-10.11 Some social networking web sites provide mechanisms for users to determine the GPS coordinates of where their friends are located at any given moment. Describe a generalization scheme that would anonymize this information using disjoint rectangles so that any reported rectangular region as a "location" always has at least k people in it, for some security parameter k.

C-10.12 Generalize the ThreeBallot system to use four ballots instead of three. What are the advantages and disadvantages of this generalization?

C-10.13 Explain why the ThreeBallot system won't achieve all of its security goals if only two ballots are used instead of three.

Projects

P-10.1 Do an experiment involving the use of additive noise for protecting a database from inference attacks. Your database should begin by generating a specific list of values that have a mean of 25.0. Then, anonymize these values by adding a random noise value, which is designed to have an expected value of 0. For instance, you could use uniformly distributed values in $[-1, 1]$ or you could use values generated by a Normal (i.e., Gaussian) distribution with mean 0. Test the degree to which the mean of your values changes as a result of this noise addition. Include tests for a list of 1,000, 10,000, and 100,000 values, and both the uniform and Normal distributions for noise.

P-10.2 Write a term paper that, based on the use of an email account that regularly gets spam, classifies and categorizes the spam this accounts gets in a given week. Categorize the spams in terms of similar goals or patterns and describe in qualitative terms the objective of the spam in each category if possible, that is, whether it is for a product, phishing attack, etc. Also describe the kind of artificial intelligence that is needed to distinguish each category of spam from real emails.

P-10.3 Write a term paper that compares and contrasts the needs of digital content providers to protect their rights to a fair compensation for the use of their work with the various restrictions possible using DRM technology. Include discussions of the conflicts of fair use and possible rights revocation.

P-10.4 Using a language like Java or Python that can process audio data, write a program that can maintain an audio library under some basic DRM functionality. Provide a way for clients to rent audio files and content owners to enforce rules for playing, expiration of playing rights, copying, etc.

P-10.5 Write a program that simulates the ThreeBallot voting scheme. Your program doesn't have to necessarily handle paper ballots, but it should have a user interface for users to vote and "take" their receipts. After all voting is done, your system should then tally and report the results in a way that people can verify their votes were counted accurately.

Chapter Notes

Griffiths and Wade describe a framework for granting and revoking permissions in a database in their seminal paper from 1976 [36]. Li, Shirani-Mehr, and Yang discuss show how to protect against inference attacks in when publishing data [55]. Signed MIME email is defined in RFC 1847. The S/MIME standard is defined in RFC 2633. An overview of DKIM is given in RFC 5585. PGP is described in the official user's guide by Zimmermann [112]. The DKIM standard is defined in RFC 4871. Kanich et ***al.*** present a study of spam conversion rates [45]. The ALPACAS system was developed by Li, Zhong, and Ramaswamy [56]. Murdoch and Anderson critique the 3D Secure authentication protocol for credit card purchases [62]. David Chaum pioneered a blind signature technique for digital cash in his 1982 paper [15]. Key graphs, which generalize key trees, are discussed by Wong, Gouda and Lam [109]. The AACS DRM specification is available on the website of the AACS Licensing Administrator (www.aacsla.com). The revocation method used in AACS is based on the work by Naor, Naor, and Lotspiech [63]. ThreeBallot and two other secure voting schemes based on paper ballots are described by Rivest and Smith [83].