Protocols for Authenticated Download to Mobile Information Appliances

Trent Jaeger *
jaegert@eecs.umich.edu
Software Systems Research Lab
EECS Department
University of Michigan
Ann Arbor, MI 48105

Aviel D. Rubin
rubin@bellcore.com
Security Research Group
Bellcore
445 South Street
Morristown, NJ 07960

Abstract

Mobile hosts download files from untrusted networks to obtain application software, information services, documents, etc. However, attackers can modify the contents of these files, either in transit or on a compromised machine. Attacks are more likely on a mobile network than a fixed network because attackers can send a malicious message to a mobile host without having to break into any machine in the network. If an attack goes undetected, the mobile host may download a file that contains: (1) erroneous or misleading data; (2) faulty applications; and (3) Trojan horses or viruses. Therefore, mobile hosts need the ability to authenticate the files they download to verify that the file downloaded is the file requested. It is difficult to authenticate files because, in general, a mobile host cannot trust any of the servers that provide location information. In this paper, we define download protocols that locate and retrieve files from an untrusted network and authenticate the downloaded file using only a single trusted principal. Energy consumption is also a concern for mobile hosts, so the protocols are designed to minimize the amount of information a mobile host will need to download.
Keywords: Mobile computing systems, file integrity, digital signatures, cryptographic digests, wide-area network file location, trusted authorities.

1 Introduction

We present a file download protocol for mobile hosts that automatically locates, retrieves, and verifies the integrity of files in an untrusted environment. Mobile hosts download files to obtain application software, information services, and various other kinds of data. For example, a mobile user may wish to download a newly available software package. Also, some mobile hosts have little or no disk space and a limited amount of RAM, so the application software used by these machines may need to be downloaded when needed. In addition, a number of the applications of mobile hosts are based on the download of files from information services on traffic, weather, and stock quotes, etc [7]. In general, file download consists of two tasks: (1) file location and retrieval and (2) file integrity verification. To improve availability, read-only files are often replicated, so a file download protocol should be able to find a nearby copy of the file. Once a mobile host retrieves a file, it should verify the integrity of the file to ensure that the file received is the one requested. Our file download protocol is designed to provide mobile hosts with automated support for both tasks.

Files are often downloaded from the Internet and used without verifying their integrity. In a fixed network, this is a very dangerous practice, but in a mobile environment, the danger is even greater. In a fixed network, a malicious attacker may become able to modify a file by: (1) compromising the server that stores the file; (2) connecting a machine to the server's LAN; or (3) compromising a machine on the network path of the file download. In a mobile environment, attacks can also be made on requests to and from the mobile host itself. For example, an attacker can transmit a maliciously tampered software package to an unsuspecting mobile host. Since the malicious software is run with the user's access rights, this software

*This work was done while this author was a summer intern at Bellcore.

can: (1) access the user's private data; (2) delete the user's data; (3) access system information, such as a password file; and (4) tie up system resources. Even worse, the malicious software may contain a virus or a Trojan horse. Bellcore's Trusted Software Integrity (Betsi) system [17] enables users to manually verify the integrity of a file obtained from the Internet. However, Betsi requires user involvement to verify that the certificate is for the file requested and to compare cryptographic digests and digital signatures to those in the certificate, so it cannot be integrated with our protocols in its current form.

File location is also a difficult problem because: (1) trust in all system components and their communications is not possible and (2) it can be expensive to find a file in a large network. Distributed file systems [2, 4, 20] and shared virtual memory systems [10, 12] define protocols for locating distributed services, but these protocols trust the servers providing location information. The problem is that machines connected to the Internet can be compromised by attackers. For example, the machines that mobile hosts use to obtain network connectivity are connected to the Internet, so trust in an arbitrary server is not advisable. Therefore, even if the location information is signed (e.g., using RSA [16]) by each machine, it cannot be trusted because one of these machines may have been compromised. Second, these distributed systems protocols are designed to make an entire name space accessible. We expect that only a small percentage of files, such as commonly-used applications and information services, need to be located dynamically, so it is not necessary to make the entire name space accessible. In a previous paper, we defined a service for downloading files in an untrusted, fixed network [9]. This protocol requires trust in only one system principal and limits the cost to locate a file, but it is not designed for a mobile network.

File download protocols must be efficient to preserve the limited energy resources of mobile hosts. It is not expected that the rate of improvement in battery lifetimes is going to keep pace with the rate of improvement in CPU performance [18], so the power available to mobile hosts will continue to be limited. Thus, any file download protocols should use as few mobile host resources as possible. The fixed network protocol described above [9] requires that the principal downloading the file control all file download and error recovery tasks. In a mobile environment, we would like to delegate some of these tasks to other processors to reduce the effort of the mobile host.

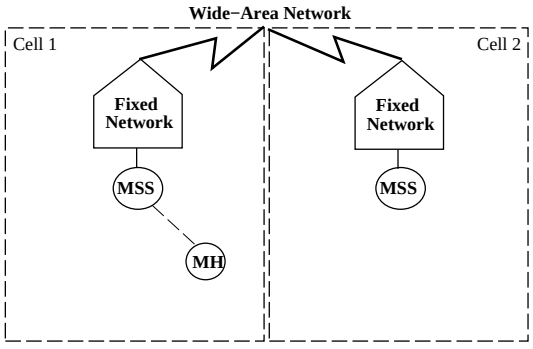

Figure 1: **Two cells in a Personal Communication Network (PCN)** – MSS: Mobile Support System; MH: Mobile Host

The goal of our service is to automate the file location, retrieval, and authentication tasks of mobile hosts. Using our protocols, a mobile host can specify the file to be retrieved and expect the service to return a copy that meets those specifications, if one is available. Files can be located and verified without requiring trust in the locating services. In fact, only one trusted principal is required for the protocol to succeed. Also, the use of mobile host resources to download location information and files is minimized.

In the next section, we describe the mobile system architecture our protocol is designed to support. In section 3, we formally define the file download problem. In section 4, we present the protocol architecture. In section 5, we detail the authenticated download protocols.

2 Mobile System Architecture

It is envisioned that a personal communication network (PCN) will be developed that will provide mobile hosts with nearly ubiquitous communication coverage [7]. The PCN (see Figure 1) will consist of a number of mobile support servers [8] (MSS's) that are attached to fixed networks and service mobile hosts in a specific area called a cell. MSS's are expected to provide mobile hosts with wide-area network connectivity. The PCN will enable a mobile host to obtain access to information services, application software, its home machine, etc.

The capabilities of mobile hosts themselves may vary markedly. At the two extremes are the "dumb terminals" and the "workstations." The dumb ter-

minals merely display information while the MSS's perform all computations for the mobile host. On the other hand, a mobile host may be a fully functional workstation with its own operating system and support software. Intermediate options include systems with CPU's that possess limited memory, such as palmtops, Tabs [21], and "terminals with the possibility of downloading some code [6]."

Examples of the types of data mobile hosts may download can be categorized as: (1) information services and (2) application software. All the types of mobile hosts described above may download information services, such as traffic, weather, stock quotes, e-mail, etc. Also, mobile workstations and limited memory CPU's may download application software for local execution. For example, a palmtop may need to download the proper application software to view data obtained from an information service. Also, a mobile workstation user may want to download new software developed within his organization. The mobile workstation user does not want to wait until he is physically connected to his organization's computers to download this software.

If the PCN is to provide truly ubiquitous connectivity, then a mobile host must be able to download a file from any MSS. File download from an arbitrary MSS without verification is dangerous, however, because of the threat of attacks against: (1) the fixed network; (2) the MSS's; and (3) the wireless communications. Attackers on the fixed network can replace files either on compromised machines or in transit. An attack may result in compromised servers responding to software requests by mobile hosts with malicious files.

Unfortunately, it is also impractical to assume that a mobile host can trust all MSS's in the PCN. The ability of a mobile host to trust an arbitrary MSS is limited by: (1) key management for MSS's and (2) the fact that MSS's may be compromised. Currently, mobile cells cover an area only 1-2 miles in diameter and are expected to decrease in size, so the number of MSS's, and hence the number of public keys, will be large. The black pages [3] service has been proposed to provide bindings of principals to keys, but use of this service costs time. In addition, MSS's are connected, at least indirectly, to the Internet, so there exists a possibility that an MSS will be compromised. Thus, even signed messages may not be trustworthy.

Even if a mobile host can trust its current MSS, attackers can also compromise transmissions between the MSS and the mobile host. Attackers with sufficient motivation can determine the transmission frequencies of the MSS (even if frequency hopping or multifrequency spread are used [14]). Therefore, attackers can arbitrarily delete, insert, and modify information destined for a mobile host without compromising the security of a single machine.

3 Problem Statement

An authenticated download protocol must solve two basic problems: (1) it must be able to locate and retrieve the file requested by the mobile host and (2) it must be able to verify that the file retrieved satisfies the mobile host's request. In this section, we formally define these two problems.

3.1 File Location and Retrieval

Before defining the file location and retrieval problem we define the following concepts:

- **Definition 1**: A *file identifier* is a tuple that includes the identifying characteristics of a file, such as its name, its author, its date, etc [1].
- **Definition 2**: A *file* is a stream of bits referenced by a file identifier.
- **Definition 3**: A *client* is a principal that requests a file using a file identifier.
- **Definition 4**: A *distribution server* is a principal that can process a file identifier and determine if it possesses a file that maps to the identifier.

When the client wants to retrieve a file, it specifies a file identifier. The file location problem is for a client to identify a distribution server that can map the file identifier to a file that it possesses. The file retrieval problem is for the client to obtain an authentic version of that file from a distribution server.

File location protocols for distributed systems can be divided into two categories: (1) pull and (2) push. Pull systems [4, 10, 12] download and cache location information on demand while push systems [2, 11, 20] use file location information that has been previously uploaded to their location servers (i.e., servers that help clients find files).

[1] A detailed definition of a file identifier is provided in Section 5.2.

Pull systems provide scalable, flexible access to a global name space, but must search for file location information on demand. However, we expect that people responsible for distribution servers will know what files they want to offer, such as in the WWW where users make file URL's available, so the search of the entire, global name space should not be necessary. Push systems are more efficient than pull systems because all the file location information is stored in the location servers, but some of these systems use techniques that are unacceptable from a security standpoint. For example, the Ameoba system [20] requires servers to be able to create daemons on client machines that handle location requests. Allowing processes to be triggered by the actions of foreign machines presents a potential security vulnerability, particularly when these processes may affect the way that the client executes future processes.

The primary security limitation of these protocols is the amount of trust that is required between the client and the servers. In each of these protocols, clients trust that the servers provide the correct location information (only the V system [4, 5] actually authenticates the servers, however). As mentioned above, even if the servers are authenticated, trust in all servers is not possible because any one of them may have been compromised. Also, a malicious attacker can modify the file identifier when it is sent from the mobile host to the MSS, so the mobile host will receive the wrong file. The mobile environment is ideally suited to such an attack.

Other mobile systems issues that these file location protocols do not address include: (1) varying mobile host location and (2) arbitrary mobile host disconnection. The location of a mobile host can change, so the file location protocols must be able to find mobile hosts and forward information to them. Also, mobile hosts will spend a great deal of time disconnected from the network. File location protocols should not require synchronous interaction with a mobile host to be completed.

3.2 File Authentication

The file authentication problem is to prove that the file retrieved meets the requirements of a client's file request. For example, a client may request a software package named f. However, if a distribution server supplies a file named f, that is not sufficient to verify to the client that the package is indeed software package f. The distribution server could have just renamed another file f.

More formally, the file authentication problem is for a client to verify that the following statements about a file f are true:

- **Statement 1**: The identity of a file f matches the identity in the file identifier provided by the client.
- **Statement 2**: The author of file f matches one of the authors in the file identifier for f.
- **Statement 3**: A cryptographic digest of file f matches a cryptographic digest of a file whose identity matches f.
- **Statement 4**: The expiration date of the cryptographic digest, identity, and author of a file f has not passed.

Statement 1 says that the retrieved file's identity matches that of the request. Statement 2 says that the file is authored by an author approved by the client. Statement 3 says that the retrieved file's contents correspond to the file contents expected. Statement 4 says that the file authentication information is current.

We assume that a client believes any statement made and digitally signed by a trusted authority. The X.509 Extended File System (XEFS) [19] permits authors to sign statements endorsing the integrity of their files, and other principals sign endorsements of other aspects of files (e.g., checked for viruses). In the XEFS, files are annotated with attributes whose values contain these endorsements, but trust in the principals endorsing the file may vary. In the Betsi system [17], a single trusted authority is defined that vouches for the integrity of the files. Therefore, a client can compare the file retrieved to authentication information signed by a trusted authority to determine if a file is authentic.

In Betsi, a trusted certification authority (CA) creates certificates that associate a registered author with a file identifier and a cryptographic digest of the file. Betsi's author registration protocol prevents an attacker from masquerading as another registered author. These certificates are digitally signed by the CA to ensure their authenticity. Therefore, a user can verify that a file is the one specified in the certificate and that its integrity is preserved.

Betsi uses the following protocol to verify file authenticity. First, a user compares the file identifier and author in the certificate to the file identifier and author expected by the user. Next, the user computes a cryptographic digest of the file (using

a one-way hash function, such as MD5 [15]) and compares this digest to the cryptographic digest in the certificate to verify the file's integrity. If the user trusts the CA and the two comparisons succeed, then the file satisfies the Betsi verification protocol. The requirement for the file to be current is implemented in Betsi via a certificate revocation list (CRL). CRL's are more difficult to implement in mobile environments because they require high availability, and mobile users may be off-line at any given time.

The Betsi system requires manual intervention to verify a file which prevents it from being integrated in a fully automated file download protocol. An automated file download protocol for fixed networks that verifies the integrity of retrieved files is described in [9]. In this protocol, a service on the client's machine controls the location process and verifies the authenticity of any downloaded files. Mobile hosts have energy limitations due to fact that they are often battery-powered. Therefore, an authenticated download protocol for mobile systems should minimize the effort of the mobile host in network operations, so it can preserve energy for computation.

4 Architecture

Our architecture is based on the following assumptions. First, each mobile host can securely obtain and store a copy of the CA's public key. An off-line mechanism must be used to distribute this key. Second, we assume that an attacker cannot certify files using another author's identity. Betsi uses an off-line mechanism to verify an author's identity during registration of an author's public key. We do not expect that Betsi's mechanism will be used for this application, but we require that some off-line verification is performed. Third, we assume that trusted software for generating cryptographic digests and for verifying digital signatures are available at each mobile host. Finally, denial-of-service attacks are generally easy to detect and hard to prevent, so we assume that they can be detected by the system and are repaired off-line.

The trust model includes the following principals:

- **Mobile Hosts**: Principals that request files
- **Mobile Support Servers (MSS's)**: Principals that provide mobile hosts with wide-area network connectivity
- **Distribution Servers**: Principals that store and distribute files
- **Location server**: Principals that map file identifiers to distribution servers
- **Authors**: Principals responsible for a file (e.g., the system administrator of a server)
- **Certification authorities (CA's)**: Principals trusted by clients to certify authors' files

Mobile hosts represent the client role in the file location problem. *Mobile support servers* (MSS's) enable mobile hosts to make file requests by providing network connectivity. Some of the tasks of the client may be replicated in the MSS to reduce the effort of the mobile host. For example, mobile support systems should try to verify the integrity and authenticity of a file before forwarding it to a mobile host. Therefore, only if the mobile host does not trust the mobile support system does it need to verify the integrity of a file, and even in most of these cases, the verification should be successful.

Distribution and location servers implement the file system for the mobile hosts. *Distribution servers* store files to be downloaded and forward these files to MSS's. *Location servers* locate distribution servers that claim to possess a valid copy of the requested file. In our architecture, the locations of all distributed files are uploaded to at least one location server, as in the push model. We also expect that location servers will maintain information about which location server administers which domains, so a location server can find a file that is not in its domain efficiently, similar to the V system [4, 5].

Certification authorities (CA's) certify the authenticity of the authors' files. An *author* sends the CA a signed message containing a file's certification information. After verifying the signature, the CA generates a certificate for the author stating that the author certified the file at the current time (see section 5.1 for the certificate definition). This contrasts with the PEM [1] model where a CA would certify the public key of each author, and authors would certify files directly. The advantages of CA's certifying files are that: (1) the certification date of the file can be trusted; and (2) certification using previously revoked public keys is prevented. The authenticity of some documents, such as legal documents, may depend on the date that they were certified. Since the CA creates the certificate containing the certification date, this date can be

trusted. Also, the CA generates all certificates, so the CA can prevent certification of files signed using a revoked key. In addition, our CA can also support implementation of certificates based on the PEM model, where appropriate.

A mobile host need only trust a single CA in our architecture. At present, we assume that each organization will have one CA. If a group of organizations wants to share information, a web of trust between the CA's of those organizations can be created using the mechanism used for PEM [1] or PGP [22] [2].

Mobile hosts need not trust any mobile support servers, location servers, or distribution servers. If an MSS or distribution server delivers an incorrect or tampered file, the client recognizes it. Any change to the certificate is detected by signature verification, and any change to the file is detected by the digest comparison. If a location server is compromised, the client can use another location server. At worst, a compromised server can cause a denial of service.

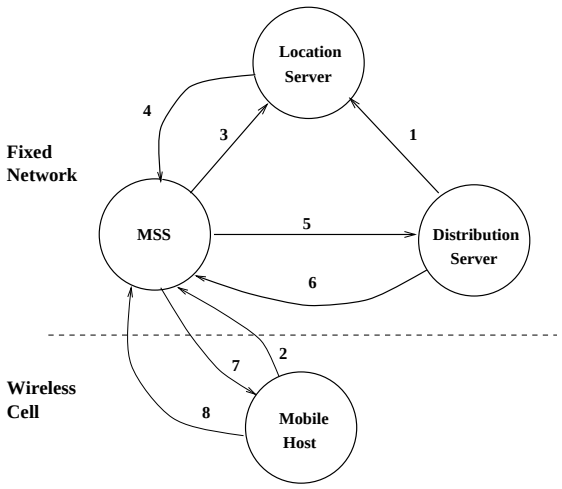

Figure 2: **Authenticated Download Protocol:** (1) distribution server publishes a file; (2) mobile host requests a file; (3) MSS forwards mobile hosts request; (4) location server returns a set of distribution servers for a file; (5) MSS requests a file from a distribution server; (6) distribution server returns a file and its certificate; (7) MSS forwards file and its certificate; and (8) a mobile host requests the file be resent by the MSS.

5 Protocol Definitions

The authenticated download protocol is shown in Figure 2. In step 1, distribution servers announce to location servers that they possess a specific file. In step 2, a mobile host sends a message requesting a file's location to an MSS. In step 3, the MSS forwards this request to a location server. In step 4, the location server returns a list of distribution servers for the file to the MSS. In step 5, the MSS chooses a distribution server and requests the specified file and its certificate. In step 6, a distribution server returns a copy of the file and its certificate to the client. The MSS may verify the authenticity of the file itself. If the verification fails, the MSS can repeat steps 5 and 6 until an authentic version of the file is retrieved. In step 7, the MSS forwards the file and its certificate to the mobile host which then verifies the file's integrity. If the mobile host's verification fails, the mobile host can request that the file be resent from the MSS (step 8). It is assumed that the mobile host can obtain network connectivity from only one MSS, so only this MSS can download the file for the mobile host.

The authenticated download protocol implements the location and retrieval of files and the verification of their integrity. The locations of the files that are available to mobile hosts are published in step 1. The remaining steps involve the transfer of file and/or location information between principals. Integrity verification is performed by both the MSS and the mobile host. The MSS verifies the integrity of the file received in step 6. Thus, if the MSS is working properly, it will always deliver a valid copy of the file to the mobile host. The mobile host still needs to verify the file it obtains in step 7 because transmission from the MSS may be forged or disrupted or the MSS may already be compromised.

This authenticated download protocol differs from the protocol for a fixed network [9] in the effort required in the mobile hosts. A client in a fixed network has the energy resources to perform repeated requests and downloads of a file. However, the number of file downloads should be minimized for mobile hosts because their energy resources are limited. In the protocol above, the MSS detects the download of false files from the fixed network. Thus, a mobile host will obtain an authentic copy of the file requested on the first request as long as the following attacks do not occur: (1) the compromise of the MSS and (2) an attack on the wireless transmission.

[2] PGP is a trademark of Philip Zimmermann.

Other problems that are unique to mobile systems are: (1) a mobile host may change MSS's and (2) a mobile host may disconnect temporarily. For example, after a mobile host requests a file from an MSS, the mobile host moves to a new cell that is administered by a different MSS. It is expected that state information about a mobile host must be transferred between MSS's when responsibility for a mobile host is transferred [7]. Therefore, information about outstanding requests should be included in this state information. If a mobile host becomes unreachable (e.g., due to shutdown) during the algorithm, the MSS will time-out the request when step 7 cannot be completed. The mobile host must then reissue the request when it comes back online. Since the MSS has already downloaded the file, the MSS can forward it to the mobile host immediately upon receipt of the request.

Integrity verification in the authenticated download protocol requires the support of cryptographic algorithms: (1) one-way hash functions and (2) digital signature algorithms. One-way hash functions generate cryptographic digests that can be used to verify a file's integrity. A one-way hash function has the following features: (1) it is a *one-way function* because it is thought to be computationally infeasible to derive the input to a calculation given the output and (2) it is a *collision-free function* because, given an input and an output, it is possible to find another input that hashes to that same output with only a negligible probability. These two features prevent an attacker from being able to modify a file, yet still obtain the cryptographic digest of the original file. In our fixed network implementation, we used the MD5 one-way hash function [15] because its code is in the public domain and it is the RFC standard one-way hash function.

Digital signatures are used to verify that the CA generated a certificate. A digital signature is created from a combination of a principal's private key and a message. The signature has the property that anyone in possession of that principal's public key can verify the integrity and authenticity of the message, and that nobody without the private key can create it. Therefore, MSS's and mobile hosts can verify that the CA created a certificate by checking the associated digital signature. In our fixed network implementation, all digital signatures are generated using the Digital Signature Algorithm (DSA) [13]. Our choice of DSA was made on the basis that NIST claims that DSA can be exported and that it is royalty-free [3]. An alternative algorithm for digital signatures is the RSA algorithm [16], but RSA has the disadvantages that its exportation is tightly controlled and its use requires royalty payments to the patent holders. However, RSA is a significantly more efficient algorithm than DSA, particularly for verification.

Below, we detail the authenticated download protocol. First, we describe the information that must be available when the protocol is initiated. Next, we describe the actions taken in each step of the protocol.

5.1 Initial Conditions

Before detailing the protocol, we identify the initial conditions assumed by the protocol. Specifically, we assume that the distribution servers have obtained copies of the files to be published and their certificates.

Registered authors certify their files with a CA trusted by those clients who will want to use the file. This enables clients to verify the contents of the file, its identifying attributes, and its certification date. The CA provides the author with an authentication certificate for the file. We assume that the distribution server obtains a copy of the file and its authentication certificate off-line.

Effectively, a file authentication certificate associates an author with a cryptographic digest of a specified file. Digests are used in the place of files because they have a small, fixed size (e.g., 128 bits for MD5). Mobile hosts can compare a digest of the file to the digest provided in the certificate to verify the integrity of the file. The integrity of the certificate is guaranteed by the signature of the trusted CA.

Other information is needed in the certificate to verify that the requested file was retrieved. Therefore, a certificate has the following fields:

- Identity of CA
- Author Name
- Author's Organization
- Author's E-Mail Address
- Name of the File
- Version Number

[3] It is currently unresolved as to whether a patent by Schnorr covers DSA, however.

- Machine
- Machine ID
- O/S
- O/S version
- Cryptographic Digest
- Expiration Date
- Latest Version?
- Date

The fields in the certificate have the following meanings. The `identity of CA` is used so the mobile host can determine which public key to use to verify the authenticity of the certificate. Author information enables the mobile host to verify the author of the file. The `version number` of the file enables the mobile host to verify that the file is the proper version. The machine and O/S information are used to verify that the file is appropriate for a specific platform. These fields are applicable to software. As described above, the `cryptographic digest` is used to verify the integrity of the file. The expiration date indicates the date when the certificate becomes invalid. The `latest version?` field specifies that the author claims that this file is the latest version at the date the certificate was created and should continue to be the latest version until the `expiration date` is reached. The `date` is the date of certification. The protocol for verifying the authenticity of a file using these certificates is detailed in the File Authentication Section below.

5.2 File Publication

Mobile hosts can locate a file on a particular distribution server because it has been 'published.' The act of 'publishing' was first used in the Ameoba system [20] as a way to advertise that a particular service resides on a particular server. In Ameoba, a file publication protocol activates server agents on client machines to catch and forward service requests. In contrast, our protocol involves upload of the location information of a file to location servers.

An important consideration in file publication is the specification of a file. A file must be specified in a way that it can be uniquely referenced. A unique file is specified using a file identifier. A file identifier consists of the following fields:

- File Name

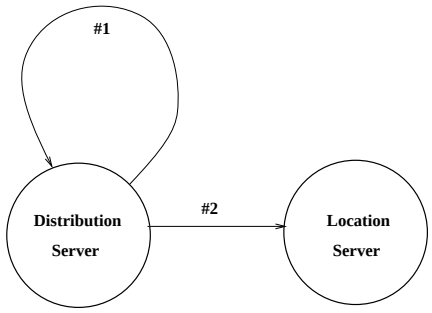

Figure 3: **The file publication protocol:** (1) Distribution server stores a new file identifier-to-pathname mapping and (2) distribution server uploads a file identifier-to-distribution server mapping to a location server.

- Author Set
- Version Number (optional)
- Machine (required for compiled software only)
- Machine version (required for compiled software only)
- O/S (required for compiled software only)
- O/S version (required for compiled software only)
- Latest Version? (optional)

A mobile host is required to know the execution platform for compiled software. For ascii files, these values are not required. If the version is not specified, then it is assumed that the `latest version?` of the file is requested.

Location servers store a map of file identifiers to the set of distribution servers that provide the associated files. Location servers should be associated with a set of distribution servers, as location servers in the V operating system [5] (called *liaison servers*) are, so a file on a specific distribution server can be found efficiently. Therefore, a distribution server publishes the file to the location server responsible for its domain.

Distribution servers store a map from a file identifier to the file pathname and the file's certificate pathname. This enables a distribution server to quickly locate a file requested.

The file publication protocol is shown in Figure 3. The protocol steps are as follows:

1. The distribution server stores a new set of file location information in its file identifier-to-pathname database. Each distribution server database entry has the following fields:
   - File Identifier
   - File Pathname
   - Certificate Pathname
2. The distribution server tells a location server to update its file identifier-to-server database. Each location server database entry has the following fields:
   - File Identifier
   - Distribution Server

The results of the file publication protocol are: (1) the location server can list the set of distribution servers that claim to possess a file that matches a file identifier and (2) the distribution server can locate that file and the file's certificate given a file identifier.

In this protocol, it is possible for attackers to publish false files on a location server. This attack will not result in a mobile host accepting a file that is not authentic, but can result in a denial-of-service if the location server database becomes full of false file locations. Attackers can be prevented from publishing false files if the location server also requires a publication message to include a valid certificate for the file. This does not prevent an attacker from publishing legitimate files in false locations, however. Off-line verification of published files by system administrators can be used to limit the number of false entries.

5.3 File Location

In the file location protocol, the MSS acts on behalf of a mobile host to obtain a set of distribution servers that claim to possess a file that matches a file identifier. If a mobile host already knows a distribution server with the desired file or the MSS has a copy of the file, then the file location protocol is not necessary.

The file location protocol is shown in Figure 4. The steps in the protocol are as follows:

1. A mobile host sends a file identifier to its current MSS. The file identifier is defined in the File Publication Section above.
2. The MSS forwards this request to its location server.

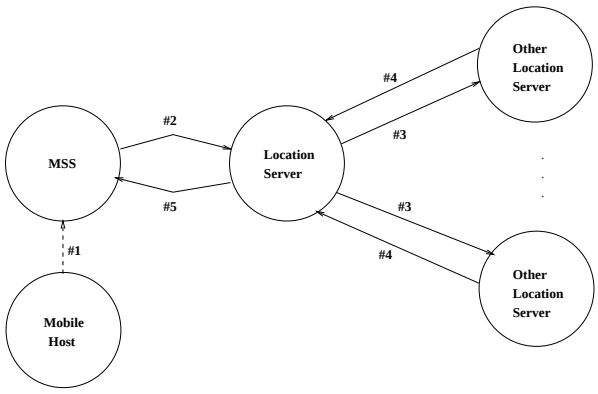

Figure 4: **The file location protocol:** (1) Mobile host sends a file identifier to its current MSS; (2) the MSS forwards this file request to its location server; (3) The MSS's location server requests that other location servers find the distribution servers that possess a file (optional); (4) Other location servers return to the MSS's location server a set of distribution servers that possess a file; (5) MSS's location server returns a set of distribution servers that possess the file to the MSS.

3. (Optional) The MSS's location server sends the file identifier to some number of the other location servers. This is only done if the MSS's location server has no knowledge of any distribution servers that possess the file.
4. (Optional) These other location servers each return a set of distribution servers to the MSS's location server.
5. The MSS's location server unions the set of distribution servers collected and returns the resultant set to the MSS.

The result is that the MSS collects a set of distribution servers (possibly empty) that its location server claims possess the requested file. We have no requirement that location servers be consistent, so if a location server has no entry that matches a file identifier then it can request the help of other location servers. Also, there is no requirement that the location server return the complete set of distribution servers possessing the file. Only one valid copy is needed. The MSS's location server caches any new information it obtains from these other location servers. Therefore, the next time a client requests the same file, the location server can provide the set of distribution servers found previously. We expect that location servers will publish new files among one another at regular increments

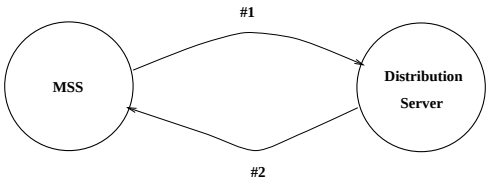

Figure 5: **The file retrieval protocol:** (1) MSS sends a file identifier to a distribution server; (2) the distribution server returns the corresponding file and its certificate.

(e.g., daily) or after a threshold number of updates have been made to them.

The main attack against file location is for an attacker to modify the location server's return messages (the location structure messages) with a false reply. Again this attack will only result in a denial-of-service, so MSS retries can overcome this attack.

5.4 File Retrieval

The file retrieval protocol enables the MSS to retrieve the file and its certificate using the set of servers obtained in the file location protocol. The file retrieval protocol is fairly straightforward, but it may need to be rerun if the file authentication fails. Because file authentication can fail, the MSS treats the file location information as a hint to the possible locations of the file. The file retrieval protocol determines how these hints are used.

The basic file retrieval protocol is shown in Figure 5. The MSS sends the same file identifier it sent to the location server in the file location protocol to a distribution server. The distribution server returns the corresponding file and its certificate. A null file indicates that the distribution server does not possess the file (or that an attack has occurred). If a file and its certificate are returned, the file authentication protocol described in the next section is run.

If file authentication fails, then the MSS must determine what the next action should be. The MSS has four options: (1) retry the retrieval using the same distribution server; (2) try the retrieval using another distribution server; (3) obtain a new distribution server set from the same location server; and (4) obtain a new distribution server set from another location server. Since we expect relatively few failures, we recommend the use of a simple, round-robin algorithm to locate a distribution server with an authentic copy of the file. First, each distribution server in the distribution server set is tried in succession. If all the distribution servers fail (the MSS can specify a limit for the number of failures), another location server is queried for a new set of distribution servers and the algorithm is repeated. If an MSS detects a failure, it notifies the server's system administrator. Verification of server state and failure repair are performed off-line.

5.5 File Authentication

When the MSS receives the file and its certificate, it should verify the authenticity of the file before forwarding it to the mobile host. This reduces the likelihood that the mobile host will receive a false or corrupted file. The file authentication protocol is used by the MSS to verify the file's authenticity. The file authentication protocol compares the file identifier generated by the MSS and the file itself to the information in the file's certificate. The result is that the MSS verifies that the file is authentic or discards the file.

The file authentication protocol consists of two steps: (1) verifying the CA's signature and (2) verifying the information in the certificate. The verification of the CA's signature is necessary to ensure that the certificate has not been modified. Any change to the certificate will cause the digital signature verification process to fail.

Once the CA's signature is verified, the information in the certificate is checked to ensure that the correct file has been retrieved. The file and its identifier are compared to the following components of the certificate:

1. Expiration Date of the Certificate
2. File Name
3. File Author
4. Version Number (optional)
5. Latest Version Number (optional)
6. Machine (optional)
7. Machine Version (optional)
8. O/S (optional)
9. O/S Version (optional)
10. Cryptographic Digest of the File

First, the expiration date of the certificate is checked. The expiration date indicates the time at which a certificate becomes invalid. An author can

use this capability to register files until a certain date. Then, the author can either re-certify the file or certify a new version of the file. This is useful for releasing updated versions of documents or software.

Next, the file identifier information (entries 2-9 above) can be verified. This is accomplished by matching the file identifier information supplied by the mobile host against the values of the file identifier attributes in the certificate. The author field verification is satisfied if the author matches one of the author's in the file identifier's `author set`.

Lastly, the integrity of the file itself is checked by computing the cryptographic digest of the file (e.g., using MD5) and comparing that value to the cryptographic digest in the certificate. If the digests match, the integrity of the software is assured with a large degree of confidence. This assurance is possible because the CA's signature guarantees that the validity of the digest and the probability that a one-way hash function computes the same digest for two distinct inputs is negligible.

Even if the hashes match, it is possible that the author may have certified malicious software in the first place. In this case, the author is directly linked to the software, so any malicious actions by the software can be attributed to the author.

5.6 Mobile Host File Authentication

Once the MSS has verified the authenticity of the file, the MSS forwards the file and its certificate to the mobile host. Even though the MSS has verified the authenticity of the file, the mobile host must re-verify the file's authenticity because: (1) the file may have been corrupted or replaced in transmission or (2) the MSS itself may have been compromised. The file authentication protocol used by the mobile host is identical to the protocol used by the MSS and described in the previous section.

If transmission is not attacked and the MSS has not been compromised, then file download to a mobile host requires minimum download effort. The mobile host sends only one request to download a file. The only data downloaded to the mobile host are the file and its certificate. No public keys need to be downloaded since the mobile hosts should already have a copy of the CA's public key. Therefore, authenticated download is optimal in the number of messages that the mobile host must process. The amount of data downloaded is minimal because the because the mobile host must download the certificate to verify the file's authenticity. If the mobile host already possesses a certificate for this file, then the mobile host should request download of the file without the certificate.

If file authentication fails, the mobile host needs to retry downloading the file. If the MSS is working properly, then the MSS should have obtained a valid copy of the file before forwarding it to the mobile host. That is, after running the file location and retrieval protocols, the MSS is a distribution server of the retrieved file for the mobile hosts in its cell. Therefore, the mobile host should repeat its request for the file to the MSS.

If several retries have failed, the mobile host can conclude that either transmission is not secure or the MSS is behaving improperly. In either case, the mobile host should terminate its requests to the MSS. The failure of this MSS should be reported to the appropriate system administrator and fixed off-line.

6 Conclusions

Mobile systems are growing in functionality and popularity. These systems give users the potential to continue working in their favorite computing environment regardless of their location. There are several technological barriers to realizing the potential of mobile computing. Of these, security is a particularly difficult challenge. Cryptographic operations are, by nature, expensive, and protocols for mobile networks are further constrained by the nature of the environment. Protocols must be optimized for low power consumption, low bandwidth, and low availability. These are not the normal constraints one thinks of when designing cryptographic protocols.

We present an architecture and protocols for authenticated download to mobile computers. These protocols guarantee that any modification to the downloaded data/software/information will be detected. Digital signatures and one-way hash functions are used to ensure that authentication certificates signed by a trusted authority can vouch for the integrity and authenticity of any file. The architecture requires one trusted authority and valid verification software for each mobile host.

Although a substantial amount of research has already been done in the area of cryptographic protocols, their application to mobile systems is new. Requirements in the areas of mobility, trust, availability, and energy efficiency complicate the design

of protocols for mobile networks. Our protocol is an example of how existing cryptographic technology can be applied to a problem in mobile computing.

References

[1] D. Balenson. Privacy enhancement for Internet electronic mail: Part III: algorithms, modes, and identifiers. Internet RFC 1423, February, 1993.

[2] A. D. Birrell, R. Levin, R. M. Needham, and M. D. Schroeder. Grapevine: An exercise in distributed computing. *Communications of the ACM*, 25(4):260–274, April 1982.

[3] M. Blaze. Transparent mistrust: OS support for cryptography-in-the-large. In *Proceedings of the 4th Workshop on Workstation Operating Systems*, pages 98–102, 1993.

[4] D. R. Cheriton. The V distributed operating system. *Communications of the ACM*, 31(3):314–333, March 1988.

[5] D. R. Cheriton and T. P. Mann. Decentralizing a global naming service for improved performance and fault-tolerance. *ACM Transactions on Computer Systems*, 7(2):147–183, May 1989.

[6] D. Goldberg and M. Tso. How to program networked portable computers. In *Proceedings of the 4th Workshop on Workstation Operating Systems*, pages 30–33, 1993.

[7] T. Imielinski and B. R. Badrinath. Mobile wireless computing: challenges in data management. *Communications of the ACM*, 37(10):18–28, October 1994.

[8] J. Ioannidis and G. Q. Maguire Jr. The design and implementation of a mobile internetworking architecture. In *Proceedings of the Winter Usenix Conference*, pages 491–502, 1993.

[9] T. Jaeger and A. D. Rubin. Preserving integrity in remote file location and retrieval. In *Proceedings of the Internet Society Symposium on Network and Distributed System Security*, 1996. To appear.

[10] E. Jul, H. Levy, N. Hutchison, and A. Black. Fine-grained mobility in the Emerald system. *ACM Transactions on Computer Systems*, 6(1):109–133, February 1988.

[11] B. Lampson. Designing a global name service. In *Proceedings of the Fifth ACM Symposium on Principles of Distributed Computing*, pages 1–10, 1986.

[12] K. Li and P. Hudak. Memory coherence in shared virtual memory systems. *ACM Transactions on Computer Systems*, 7(4):321–359, November 1989.

[13] NIST FIPS PUB XX. Digital Signature Standard. National Institute of Standards and Technology, U.S. Department of Commerce, February 1993. DRAFT.

[14] M. Nemzow. *Implementing Wireless Networks*. McGraw-Hill, 1995.

[15] R. Rivest. The MD5 message digest algorithm. Internet RFC 1321, April, 1992.

[16] R. Rivest, A. Shamir, and L. Adleman. On digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[17] A. D. Rubin. Trusted distribution of software over the internet. In *Proceedings of the Internet Society Symposium on Network and Distributed System Security*, 1995.

[18] S. Sheng, A. Chandrasekaran, and R. W. Brodersen. A portable multimedia terminal for personal communications. *IEEE Communications*, pages 64–75, December 1992.

[19] R. K. Smart. The X.509 extended file system. In *Proceedings of the 1994 Internet Society Symposium on Network and Distributed System Security*, pages 129–137, February 1994.

[20] A. S. Tannenbaum, R. van Renesse, H. van Staveren, G. Sharp, S. Mullender, J. Jansen, and G. van Russom. Experiences with the Ameoba distributed operating system. *Communications of ACM*, 33(12):46–63, December 1990.

[21] M. Weiser. The computer for the twenty-first century. *Scientific American*, 265(3):94–104, September 1993.

[22] P. Zimmermann. PGP user's guide. Distributed by the Massachusetts Institute of Technology, May 1994.