Authentication and Secure Communication

Jeff Chase
Duke University

The First Axiom of Security

- "Security is at least as much a social problem as it is a technical problem."
  - Translation: humans are the weak link.
- We will focus on the technical elements, but do not lose sight of the social dimension.
  - Keys left in lock
  - Phishing
  - Executable attachments
  - Trojan software
  - Post-it passwords
  - Bribes, torture, etc.
  - Etc.

Exhibit A

This is a picture of a $2.5B move in the value of Emulex Corporation, in response to a fraudulent press release by short-sellers through InternetWire in 2000. The release was widely disseminated by news media as a statement from Emulex management, but media failed to authenticate it.

"Humans are incapable of securely storing high-quality cryptographic keys, and they have unacceptable speed and accuracy when performing cryptographic operations. (They are also large, expensive to maintain, difficult to manage, and they pollute the environment.) It is astonishing that these devices continue to be manufactured and deployed. But they are sufficiently pervasive that we must design our protocols around their limitations."

- Kaufman, Perlman, and Speciner

Trusted vs. Trustworthy (NSA)

- Trusted
  - A component that can break the security policy if it fails. ("It has power")
  - Integrity cannot be verified by external observation. ("You can’t tell if it breaks")
- Trustworthy
  - A component that is unlikely to fail.
- Trusted Computing Base (TCB)
  - The minimal core of a computer system that is trusted, and so must be trustworthy if the system is to remain safe.

As quoted in:

[Image of a chart or graph related to the content]
Questions and Answers #1

- Who is the sender?
  - Authentication
- Is the sender allowed to do this?
  - Authorization
- Is this really what the sender said?
  - Integrity
- Could anyone else have intercepted it?
  - Privacy

Questions and Answers #2

- Authentication?
  - Challenge/response: passwords, certificates
    - A subject bound to a strong identity is a principal.
- Authorization?
  - Access control lists or capabilities (ticket/token)
- Integrity?
  - Message digests and digital signatures
- Privacy
  - Encryption (provides integrity too)

All of these require some form of a shared secret or shared trust in a third party, or both.

Cryptography

- Secret key (e.g., DES)
- Public key (e.g., RSA)
- Message digest (e.g., MD5)

Security

- Authentication
- Privacy
- Integrity

Familiar names for the protagonists in security protocols

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>First participant</td>
</tr>
<tr>
<td>Bob</td>
<td>Second participant</td>
</tr>
<tr>
<td>Carol</td>
<td>Participant in three- and four-party protocols</td>
</tr>
<tr>
<td>Dave</td>
<td>Participant in four-party protocols</td>
</tr>
<tr>
<td>Eve</td>
<td>Eavesdropper</td>
</tr>
<tr>
<td>Mallory</td>
<td>Malicious attacker</td>
</tr>
<tr>
<td>Sara</td>
<td>A server</td>
</tr>
</tbody>
</table>

Cryptography for Busy People

- Encrypt and Decrypt functions
  - M = Decrypt(Encrypt(M))
  - Standard and efficient enough to be practical.
- Crypto functions are parameterized by keys.
  - Fixed-width “random” value.
  - Everybody has their own key(s) or key pair(s).
  - “Computationally infeasible” to decrypt without the key.
  - Key length really matters.
- Two fundamental variants:
  - Public-key or asymmetric crypto (e.g., RSA)
  - Secret-key, private-key, symmetric crypto (e.g., DES)

Using Crypto: the Basics

- Privacy
  - Attacker cannot read encrypted data.
- Integrity
  - Encrypt a hash/checksum/digest of the message.
- Authentication
  - Challenge-response with a nonce
    - "number used once"
  - Receiver encrypts the nonce and sends it back.
  - Proves it possesses the matching key.
  - Nonces can be timestamps, serial numbers, etc. to prevent replay attacks.
Symmetric Crypto
- "Secret key" or "private key" cryptography.
- DES, 3DES, DESX, IDEA, AES
- Sender and receiver must possess a shared secret
  - Shared key K
- Message M, Key K
  \[\{M\}_K = \text{Encrypt}(M, K)\]
  \[M = \text{Decrypt}(\{M\}_K, K)\]

Asymmetric Crypto
- Sometimes called "public key" cryptography.
- Each subject/principal possesses a keypair: \(K\) and \(K^{-1}\)
  - \(\text{Decrypt}(K, \text{Encrypt}(K^{-1}, M)) = M\)
  - Given \(\text{Encrypt}(K^{-1}, M)\), cannot compute \(M\) without \(K\).
  - Given \(M\) and \(\text{Encrypt}(K^{-1}, M)\), cannot compute \(K\) or \(K^{-1}\)
  - Given \(x\), cannot compute \(y\) such that \(\text{Decrypt}(K, y) = x\), unless you know \(K^{-1}\).
- Each principal keeps one key private.
- The inverse key may be public.
- Either key can be used to encrypt/decrypt.

Pros and Cons
Symmetric crypto (DES, AES, ...)
  - **Pros:** cheap and easily supported by hardware
  - **Cons:** need a shared secret.
    - Shared secrets are harder to keep secret.
    - key distribution problem
Asymmetric crypto (Diffie-Hellman, RSA)
  - **Cons:** expensive
  - **Pros:** no need for a shared secret
    - The recipient just needs to know sender’s public key.
    - Solves the private-key distribution problem
  - **Cons:** introduces a new public-key distribution problem

Figure 7.3
Cryptography notations

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(K_A)</td>
<td>Alice’s secret key</td>
</tr>
<tr>
<td>(K_B)</td>
<td>Bob’s secret key</td>
</tr>
<tr>
<td>(K_{AB})</td>
<td>Secret key shared between Alice and Bob</td>
</tr>
<tr>
<td>(K_{priv})</td>
<td>Alice’s private key (known only to Alice)</td>
</tr>
<tr>
<td>(K_{pub})</td>
<td>Alice’s public key (published by Alice for all to read)</td>
</tr>
<tr>
<td>({M}_K)</td>
<td>Message encrypted with key (K)</td>
</tr>
<tr>
<td>([M]_K)</td>
<td>Message signed with key (K)</td>
</tr>
</tbody>
</table>

Performance of encryption and secure digest algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Key size/Hash size (bits)</th>
<th>Extrapolated/PRB optimized speed (b/ops/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEA</td>
<td>128</td>
<td>700</td>
</tr>
<tr>
<td>DES</td>
<td>56</td>
<td>350</td>
</tr>
<tr>
<td>Triple-DES</td>
<td>112</td>
<td>120</td>
</tr>
<tr>
<td>IDEA</td>
<td>128</td>
<td>700</td>
</tr>
<tr>
<td>RSA</td>
<td>512</td>
<td>7</td>
</tr>
<tr>
<td>RSA</td>
<td>2048</td>
<td>1</td>
</tr>
<tr>
<td>MD5</td>
<td>128</td>
<td>1740</td>
</tr>
<tr>
<td>SHA</td>
<td>160</td>
<td>570</td>
</tr>
</tbody>
</table>

Better Together, Part 1
- Use asymmetric crypto just to "handshake" and establish a secret session key.
- Converse with the efficiency of symmetric crypto.
  - **Example:** Secure Sockets Layer (SSL) or Transport-Layer Security (TLS), used in HTTPS.
- End-to-end security above TCP.

```
"SYN, etc."
"My public key is K."
"Let's establish a session key \(\{S\}_K\)."
```
SSL is not so simple...

- How do we know who we are talking to?
- Do we care? Somebody does...
- How do we prevent replays of encrypted content?
- SSL/TLS uses this basic handshake protocol, but there are many other aspects:
  - Nonces, serial numbers, timestamps
  - Hashes and MACs
  - Certificates
  - More on this later...

Secure Hash / Message Digest

- Well-known, standard, hash functions digest = h(M).
  - MD5, SHA1
  - Very efficient to compute.
  - Digest is a small, fixed-width quantity: i.e., it is a hash.

  - Often called a fingerprint or cryptographic checksum.
    - Collision-resistant
      - There exist distinct M1 and M2 such that h(M1) == h(M2).
      - Such collisions are "hard" to find.
    - One way
      - Given digest, cannot generate an M with h(M) == digest.
    - Secure
      - The digest does not help to discover any part of M.

Messages with both Authenticity and Secrecy

- How does A send a message x to B with:
  - Authenticity (B knows that only A could have sent it)
  - Secrecy (A knows that only B can read the message)

1. A Transmits the following message x:
   - ((x)KA)KB

2. What if x is large (performance concerns)?
   - A transmits KA to B, B transmits KB to A
   - A picks JA, transmits {JA}KB to B
   - B picks JB, transmits {JB}KA to A
   - Each computes secret key, Ksk = Hash(JA, JB)
   - A transmits {x}Ksk to B

[Validat]

Digital Signatures

- How can the sender/writer A of M allow any receiver to verify or prove that A sent/stored M?
  - Authenticity

- Digital signature using asymmetric crypto, e.g., RSA.
  - A computes digest h(M)
  - A computes signature {h(M)}K, and appends to M.
  - Encrypted with A’s private key K.
  - Receiver decrypts digest using A’s public key K-1
  - Receiver computes h(M) and compares to digest.

- Digital signatures are "unforgeable" and "non-repudiable".
  - Unlike physical signatures, they are bound to a particular message or document.
  - Legally binding in the US.

Digital signatures with public keys

[Diagram]

Low-cost signatures with a shared secret key

[Diagram]
Public Key Distribution

- The "key" challenge today is public key distribution (and revocation).
- **Approach #1**: trust e-mail/web (i.e., assume DNS and IP really go where you want, and authenticate the source.)
  - Example: PGP, GPG, "pretty good"… or do it in person.
- **Approach #2**: use a Public Key Infrastructure (PKI)
  - Requires everyone to agree on a central point of trust (Certifying Authority or CA).
  - Difficult to understand and deploy.
  - Hierarchy helps.
- **Approach #3**: "web of trust" in which parties establish pairwise trust and endorse public keys of third parties.
  - Local example: SHARP. Involves transitive trust.

Certifying Public Keys

- Digital signatures enable any entity to endorse the (public key, identity) binding of another entity.
- A certificate is a special type of digitally signed document:
  - "I certify that the public key in this document belongs to the entity named in this document, signed X."
  - Recipient must trust the issuer X, and must know the public key of X.
  - E.g., X may be a widely trusted certifying authority (CA) whose public key is widely available.
    - The public key of Verisign is wired into every browser.

Fig. 7.13

X509 Certificate format

<table>
<thead>
<tr>
<th>Subject</th>
<th>Distinguished Name, Public Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issuer</td>
<td>Distinguished Name, Signature</td>
</tr>
<tr>
<td>Period of validity</td>
<td>Not Before Date, Not After Date</td>
</tr>
<tr>
<td>Administrative information</td>
<td>Version, Serial Number</td>
</tr>
<tr>
<td>Extended Information</td>
<td></td>
</tr>
</tbody>
</table>

Certificate Hierarchy or Web of Trust

- Chain of Trust
  - If X certifies that a certain public key belongs to Y, and Y certifies that another public key belongs to Z, then there exists a chain of certificates from X to Z.
  - Someone that wants to verify Z’s public key has to know X’s public key and follow the chain.
  - X forms the root of a tree (web?)
- Certificate Revocation List
  - What happens when a private key is compromised?

PKI

- Public Key Infrastructure
  - Everyone trusts some root CAs.
    - Sure...
  - Institutions/organizations set up their own CAs, and the root CAs endorse them to issue certificates for their members.
    - $$$
  - And so on, recursively, to form a hierarchy like DNS.
- Network applications will have access to the keypairs and certificates of their users, and will validate the certificates of servers.
  - Any day now...

What happens…

https://www.consumefest.com/shop.html
Secure HTTP

- Uses SSL/TLS over TCP.
- Browser has some set of public keys for root CAs wired into it.
- Browser always authenticates the server.
  - Server presents certificate signed by root CA.
  - Domain name must match the certificate, etc.
- Server optionally requests to authenticate the browser.
  - Browser presents certificate.
  - Password authentication is much more common.
- Browser and server negotiate a bulk cipher and secret session key.

A Short Quiz

1. What is the most important advantage of symmetric crypto (DES) relative to asymmetric crypto (RSA)?
2. What is the most important advantage of asymmetric crypto relative to symmetric crypto?
3. What is the most important limitation/challenge for asymmetric crypto with respect to security?
4. Why does SSL "change ciphers" during the handshake?
5. How does SSL solve the key distribution problem for symmetric crypto?
6. Is key exchange vulnerable to man-in-the-middle attacks?

Handshake Protocol Structure

![Handshake Diagram](image)

Figure 7.17 SSL protocol stack

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key exchange method</td>
<td>the method to be used for exchange of a session key</td>
<td>RSA with public-key certificates</td>
</tr>
<tr>
<td>Cipher for data transfer</td>
<td>the block or stream cipher to be used for data</td>
<td>IDEA</td>
</tr>
<tr>
<td>Message digest function</td>
<td>for creating message authentication codes (MACs)</td>
<td>SHA</td>
</tr>
</tbody>
</table>

Figure 7.18 SSL handshake protocol

![Handshake Diagram](image)

Figure 7.19 SSL handshake configuration options

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key exchange method</td>
<td>the method to be used for exchange of a session key</td>
<td>RSA with public-key certificates</td>
</tr>
<tr>
<td>Cipher for data transfer</td>
<td>the block or stream cipher to be used for data</td>
<td>IDEA</td>
</tr>
<tr>
<td>Message digest function</td>
<td>for creating message authentication codes (MACs)</td>
<td>SHA</td>
</tr>
</tbody>
</table>
OpenSSH

- Uses SSL
  - User's public key installed on host side
  - Host's public key installed on client side
- Or Kerberos

SSL Questions

- How do SSL endpoints verify the integrity of certificates (IDs)?
- Does s-http guarantee non-repudiation for electronic transactions? Why/how or why not?
- Does SSL guarantee security of (say) credit numbers in electronic commerce?

More PKI

- (Public key) infrastructures
  - Many organizations now have set up their own
  - Many have not (e.g., Duke)
- Public (key infrastructure)
  - Still elusive
  - Failure of Secure Electronic Transactions (SET)

"Using encryption on the Internet is the equivalent of arranging an armored car to deliver credit-card information from someone living in a cardboard box to someone living on a park bench."

- Gene Spafford, CERIAS @ Purdue
PGP

- Pretty Good Privacy
- Each user has an asymmetric keypair
- Secure e-mail, possibly with multiple receivers
  - Digitally sign message with your private key.
  - Encrypt message and signature with random session key.
  - Append session key encrypted with public key of each intended recipient.
- Users may sign/endorse each other’s public keys and endorsements.
- Should this be illegal?
  - Zimmerman case, 1993

Kerberos 101

- Secure end-to-end communication (like SSL)
  - But always authenticates both ends
- Trusted authentication server (like SSL)
  - But requires synchronous interaction with AS
- Symmetric crypto only
  - No RSA, no certificates, no PKI.
  - (Actually, webauth uses a certificate to authenticate the authentication server.)
- A form of single sign-on
  - Only have to type your password to the AS
- Based on “Needham-Schroeder key distribution”

Simple shared-secret based cryptographic authentication

![Diagram of Kerberos authentication process]

Add mutual authentication

![Diagram of mutual authentication process]

Problems with this scheme

- Generalizing the model for m users and n services, requires a priori distribution of m x n shared keys
- Possible improvement:
  - Use trusted 3rd party, with which each user and service shares a secret key: m + n keys
  - Also has important security advantages

https://www.library.duke.edu
Mediated Authentication

- A trusted third party mediates authentication
- Called the Key Distribution Center (KDC)
  - aka Authentication Server
- Each user and service shares a secret key with the KDC
- KDC generates a session key, and securely distributes it to communicating parties
- Communicating parties prove to each other that they know the session key

Kerberos (almost)

Kerberos (roughly)
Kerberos (detailed)

- Each user and service registers a secret key with the KDC
- Everyone trusts the KDC
  - "Put all your eggs in one basket, and then watch that basket very carefully" - Anonymous Mark Twain
- The user’s key is derived from a password, by applying a hash function
- The service key is a large random number and stored on the server

Mediated Authentication

- Nomenclature:
  - K_a = Master key for “alice”, shared by alice and the KDC
  - K_ab = Session key shared by “alice” and “bob”
  - T_b = Ticket to use “bob”
  - K[data] = “data” encrypted with key “K”

Don’t Forget

1. All of this relies on various fragile assumptions about people and communities.
   - Security technology only works if people use it.
   - Find the weakest link in the end-to-end chain.
   - Compromised key? All bets are off.
   - Beware false sense of security! (E.g., WEP)
2. Design for easy, incremental, organic deployment.
   - What layer? IPSEC or VPN vs. TLS
3. Understand full range of potential attacks.
   - Man-in-middle, replays and nonces, challenge/response
   - Useful model to guide analysis: logic of “belief” (BAN)

SSL record protocol

PKI: The Concept

VeriSign (or Thawte, etc.) issues certificate signing keys to organizations.