CSc 466/566

Computer Security

2 : Introduction — Mechanisms

Version: 2012/01/30 15:50:36

Department of Computer Science
University of Arizona

collberg@gmail.com

Christian Collberg

Outline

Security Principles

How do we build secure computing systems? There are 10 well-known principles:

1. Economy of mechanisms.
2. Fail-safe defaults.
3. Complete mediation.
4. Open design.
5. Separation of privilege.
6. Least privilege.
7. Least common mechanism.
8. Psychological acceptability.
9. Work factor.
10. Compromise recording.

Security Principles: Economy of mechanisms

Definition (Economy of mechanisms)

Keep the design and implementation as simple and small as possible.

- Good engineering principle in general!
- Necessary in order to effectively inspect and analyze software for security vulnerabilities, such as reading code line-by-line.

Security Principles: Fail-safe defaults

Definition (Fail-safe defaults)

The default security configuration should be conservative.

- The default situation for a computer system should be to *not* have access.
- The protection scheme should list those conditions under which access is permitted.
- Examples:
  1. When adding a new user to the system, he should have minimal access to files and other resources.

Security Principles: Fail-safe defaults

Definition (Fail-safe defaults)

The default security configuration should be conservative.

- The default situation for a computer system should be to *not* have access.
- The protection scheme should list those conditions under which access is permitted.
- Examples:
  1. When adding a new user to the system, he should have minimal access to files and other resources.
  2. By default, when downloading code from the web, it should not be directly executable.

Security Principles: Complete mediation

Definition (Complete mediation)

Every time a resource is accessed, the access should be checked against a protection scheme.

- Don't cache the results of previous security checks!
- Examples:
  1. Your bank logs you out and asks you to log back in ever 15 minutes.

Security Principles: Complete mediation

Definition (Complete mediation)

Every time a resource is accessed, the access should be checked against a protection scheme.

- Don't cache the results of previous security checks!
- Examples:
  1. Your bank logs you out and asks you to log back in ever 15 minutes.
  2. Unix's `sudo` command allows you to issue several commands after an initial authorization, but after a period of time, you have to re-enter your superuser password.

Security Principles: Complete mediation

Definition (Complete mediation)

Every time a resource is accessed, the access should be checked against a protection scheme.

- Don't cache the results of previous security checks!
- Examples:
  1. Your bank logs you out and asks you to log back in ever 15 minutes.
  2. Unix's `sudo` command allows you to issue several commands after an initial authorization, but after a period of time, you have to re-enter your superuser password.
  3. A program checks permissions on a file only the first time it opens it — what happens if the permissions change later while the program is still running?

Security Principles: Open design

Definition (Open design)

The security architecture and design of a system should be made publically available.

- Only cryptographic keys should be kept secret!
- Open design: Allows multiple parties to examine a system for vulnerabilities.
- Open implementation (open source): Anyone can find and fix bugs.
- Opposite: security-through-obscurity.
- Examples:
  1. Cryptographic algorithms which are safe only if kept secret — once broken, hard to update! Keys are easier to replace if compromised.

Security Principles: Open design. . .

Six design principles for military ciphers (Auguste Kerckhoffs, *La Cryptographie Militaire*, 1883):

1. The system must be practically, if not mathematically, indecipherable;
2. It must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience;
3. Its key must be communicable and retainable without the help of written notes, and changeable or modifiable at the will of the correspondents;
4. It must be applicable to telegraphic correspondence;
5. It must be portable, and its usage and function must not require the concourse of several people;
6. Finally, it is necessary, given the circumstances that command its application, that the system be easy to use, requiring neither mental strain nor the knowledge of a long series of rules to observe.

Security Principles: Separation of privilege

Definition (Separation of privilege)

Multiple conditions should be needed to access a resource.

- Also: components of a system should be separated so that a security breach of one won't affect another.
- Examples:
  1. Two keys to open a safe-deposit box.

Security Principles: Separation of privilege

Definition (Separation of privilege)

Multiple conditions should be needed to access a resource.

- Also: components of a system should be separated so that a security breach of one won't affect another.
- Examples:
  1. Two keys to open a safe-deposit box.
  2. Two commands to launch an intercontinental ballistic missile.

Security Principles: Least privilege

Definition (Least privilege)

Users and processes should operate with no more privileges than they need to function properly.

- Limits the damage if an application or account is compromised.
- Examples:
  1. The military's need-to-know principle.

Security Principles: Least privilege

Definition (Least privilege)

Users and processes should operate with no more privileges than they need to function properly.

- Limits the damage if an application or account is compromised.
- Examples:
  1. The military's need-to-know principle.
  2. Code injected into a web browser can do more damage if the browser runs as superuser — the browser should instead run with minimal privileges.

Security Principles: Least common mechanism

Definition (Least common mechanism)

Multiple users shouldn't share the same mechanism to access a resource.

- Shared mechanisms mean channels that could transmit information, leading to unwanted information paths between users.
- Examples:
  1. If two users accesses the same file, they should do so using different channels.

Security Principles: Psychological acceptability

Definition (Psychological acceptability)

User interfaces should be intuitive and security settings should be set to what a user might reasonably expect.

- Examples:
  1. Why don't we always encrypt all email? Apparently it is difficult to design intuitive interfaces.

Security Principles: Work factor

Definition (Work factor)

The cost of circumventing a security mechanism should be compared to the resources available to the attacker.

- Examples:
  1. Protecting student grades: most students probably aren't very accomplished hackers.

Security Principles: Work factor

Definition (Work factor)

The cost of circumventing a security mechanism should be compared to the resources available to the attacker.

- Examples:
  1. Protecting student grades: most students probably aren't very accomplished hackers.
  2. Protecting military secrets: the adversary is a nation state with unlimited resources.

Security Principles: Work factor

Definition (Work factor)

The cost of circumventing a security mechanism should be compared to the resources available to the attacker.

- Examples:
  1. Protecting student grades: most students probably aren't very accomplished hackers.
  2. Protecting military secrets: the adversary is a nation state with unlimited resources.
  3. Brute force password cracking: now feasible with more powerful computing systems.

Security Principles: Work factor

Definition (Work factor)

The cost of circumventing a security mechanism should be compared to the resources available to the attacker.

- Examples:
  1. Protecting student grades: most students probably aren't very accomplished hackers.
  2. Protecting military secrets: the adversary is a nation state with unlimited resources.
  3. Brute force password cracking: now feasible with more powerful computing systems.

Security Principles: Work factor

Definition (Work factor)

The cost of circumventing a security mechanism should be compared to the resources available to the attacker.

- Examples:
  1. Protecting student grades: most students probably aren't very accomplished hackers.
  2. Protecting military secrets: the adversary is a nation state with unlimited resources.
  3. Brute force password cracking: now feasible with more powerful computing systems.
- Hard to determine work factor if the attacker can get help from automating the attack.

Security Principles: Compromise recording

Definition (Compromise recording)

Logging a security breach may be more effective than protecting against it.

- Assumes the attacker can't erase logs to hide their breach.
- Examples:
    1. Surveillance cameras to detect but not prevent crime.

Security Principles: Compromise recording

Definition (Compromise recording)

Logging a security breach may be more effective than protecting against it.

- Assumes the attacker can't erase logs to hide their breach.
- Examples:
  1. Surveillance cameras to detect but not prevent crime.
  2. Access logs of security sensitive files.

In-Class Exercise I — Goodrich & Tamassia R-1.16

- Give an example how someone might use security-by-obscurity in the design of a system and what the consequences could be.

Outline

Access Control Models

- We should determine who has the right to access to a piece of information.
- If we can control access to information, we can prevent attacks against confidentiality, anonymity, and integrity.
- Someone (eg. system administrators) should restrict access to those who should have access: they should apply the principle of least privilege.

Access Control Matrices

Definition (subject)

User, group, or system that can perform actions.

Definition (object)

File, directory, document, device, resource for which we want to define access rights.

- An access control matrix is a table defining permissions: rows are subjects, columns objects.
- Each table cell holds the rights of the subject to access the object.
- Rights: read, write, copy, execute, delete, annotate, ...

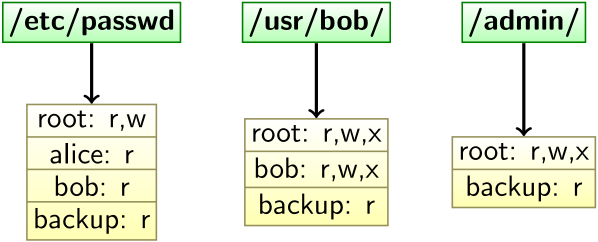
Access Control Matrices: Example

	/etc/passwd	/usr/bob/	/admin/
root	read, write	read, write, execute	read, write, execute
alice	read		
bob	read	read, write, execute	
backup	read	read, execute	read, execute

- Advantages: fast access
- Disadvantages: size

Access Control Lists (ACLs)

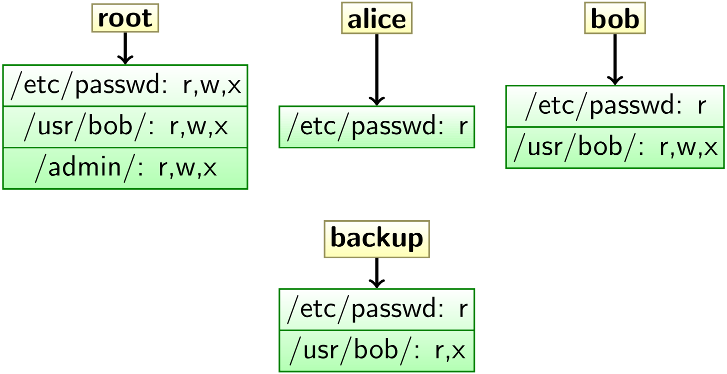
- Is object-centered: for every object *o* list (only) the subjects *s* that have access to *o*, and *s*' access rights.



- Advantages: size, *o*'s ACL can be stored directly as *o*'s metadata
- Disadvantages: can't enumerate a subject's rights (for example when a subject is removed from the system).

Capabilities

- Is subject-centered: for every subject s list (only) the objects o that s has non-empty access to, and o's access rights.



Capabilities...

- Advantages:
  1. size,
  2. easy to enumerate a subject's rights (for example when a subject is removed from the system),
  3. easy to check of subject s can access object o.
- Disadvantages: can't enumerate who has access to an object o.

Role-Based Access Control (RBAC)

- In Role-Based Access Control we replace subjects by roles in any of the access control data structures.
- Each role gets the appropriate access rights.
- Subjects are assigned to roles
- Examples:
  1. CS department roles: faculty, student, sysadmins, department head, TA, ...
  2. CS department subjects: bob={student,TA}, alice={faculty}, wendy={faculty,department head}
- A subject's access rights is the union of the rights of its various roles.

Role Hierarchies

- In a role hierarchy access rights propagate *up* the hierarchy: a node *n* inherits all the rights of it's children.
- Examples:
  1. CS department: The bottom role is member; above member is faculty and student who both inherit the rights of member; above faculty is department head who inherits the rights of faculty.
- Advantages: fewer rules since there are fewer roles than subjects.
- Disadvantages: not implemented in current operating systems.

In-Class Exercise

- Users: Alice, Bob, Cyndy.
- Alice owns `alicerc`, Bob and Cyndy can read it.
- Bob owns `bobrc`, Cyndy can read and write it, Alice can read it.
- Cyndy owns `cyndyrc`, only she can read and write it.

1. Create the access control matrix.
2. Cindy lets Alice read `cyndyrc`. Alice no longer allows Bob to read `alicerc`. Show the new matrix.

Source: Bishop, *Introduction to Computer Security*.

Outline

Cryptographic Concepts

- Cryptography underlies many of the technical means for enforcing security policies.
- Traditionally, encryption is modeled as two parties Alice and Bob who are communicating over an insecure link. An eavesdropper, Eve, is listening in to their communication.

Terminology: Encryption

Definition (encryption)

Disguising a message to hide its contents.

Definition (plaintext)

A message we want to transfer securely (aka. cleartext).

Definition (ciphertext)

The encrypted form of the plaintext message.

Definition (encipher)

Converting the plaintext to the ciphertext. Also encrypt.

Definition (decipher)

Converting the ciphertext to the plaintext. Also decrypt.

Terminology: Encoding

Definition (encode)

Converting the plaintext into a standard alphabet.

Definition (decode)

Converting the encoded message back into the plaintext.

- Example: uuencode converts a binary file into ASCII text.

```
> echo hello | uuencode -m -o outfile remotefile
> cat outfile
begin-base64 644 remotefile
aGVsbG8K
====
> uudecode -p outfile
hello
```

Terminology: Ciphers

Definition (cipher)

A map from the space of the plaintext to the space of the ciphertext. Also cypher.

Definition (stream cipher)

A cipher that enciphers the plaintext one character at a time.

Definition (block cipher)

A cipher that enciphers the plaintext in chunks of characters.

- A cipher is thus an algorithm for encryption/encipherment.
- Examples: RSA, DES, ...

Mathematical Notation

- Common abbreviations:
  - P: The plaintext.
  - M: The plaintext (message).
  - C: The ciphertext.
  - E: The encryption function.
  - D: The decryption function.
- The encryption/decryption process:

$$
\begin{aligned}
E(M) &= C \\
D(C) &= M \\
D(E(M)) &= M
\end{aligned}
$$

- It should be safe to transmit C over an insecure channel since the ciphers are chosen such that it is infeasible for anyone but Alice and Bob to find M given C.

Keys

- Ciphers need some sort of secret information known only to Alice and Bob. This is the key.
- Mathematical notation:

  K: The key, used by the encryption and decryption functions.

$$
\begin{aligned}
E_K(M) &= C \\
D_K(C) &= M \\
D_K(E_K(M)) &= M
\end{aligned}
$$

Definition (keyspace)

The range of possible values of the key.

Symmetric-key vs. Public-key Algorithms

Definition (Symmetric-key Algorithms)

Symmetric-key cryptographic algorithms use identical keys for encryption and decryption.

Definition (Public-key Algorithms)

Public-key cryptographic algorithms use different keys for encryption and decryption.

$$\begin{aligned} E_K(M) &= C \\ D_K(C) &= M \\ D_K(E_K(M)) &= M \end{aligned}$$

$$\begin{aligned} E_{K_1}(M) &= C \\ D_{K_2}(C) &= M \\ D_{K_2}(E_{K_1}(M)) &= M \end{aligned}$$

Cryptosystems

- To be able to communicate using ciphers we need
    1. Set of possible plaintexts
    2. Set of possible ciphertexts
    3. Set of encryption keys
    4. Set of decryption keys
    5. Correspondence between encryption and decryption keys
    6. Encryption algorithm
    7. Decryption algorithm
- This is known as a cryptosystem.

Monoalphabetic Substitution Ciphers: Caesar Cipher

- Add 3 to the ASCII value of each character, mod 26:

$$A \rightarrow D, B \rightarrow E, X \rightarrow A, \ldots.$$

- Cryptosystem:
  - Set of possible plaintexts and ciphertexts: Latin alphabet
  - Set of encryption keys = $\{3\}$
  - Set of decryption keys = $\{$-3$\}$
  - Decryption key = - Encryption key
  - Encryption algorithm = Decryption algorithm = $(x + \mathrm{key})$ mod 26.

Monoalphabetic Substitution Ciphers: ROT13

- Unix utility used on Usenet. Adds 13 mod 26 to each letter.

$$P = \mathtt{ROT13}(\mathtt{ROT13}(P)).$$

```
> echo "hello" | tr 'A-Za-z' 'N-ZA-Mn-za-m'
uryyb
> echo "uryyb" | tr 'A-Za-z' 'N-ZA-Mn-za-m'
hello
```

Symmetric-key Encryption Protocol

- Assuming that we have access to a symmetric-key cryptosystem (DES is an example), how do we use it?
- We have to describe a protocol that shows how each party uses the cryptosystem to solve a communication/security problem.

Symmetric-key Encryption Protocol

- Assuming that we have access to a symmetric-key cryptosystem (DES is an example), how do we use it?
- We have to describe a protocol that shows how each party uses the cryptosystem to solve a communication/security problem.

1. Alice and Bob agree on a cryptosystem.

Symmetric-key Encryption Protocol

- Assuming that we have access to a symmetric-key cryptosystem (DES is an example), how do we use it?
- We have to describe a protocol that shows how each party uses the cryptosystem to solve a communication/security problem.

1. Alice and Bob agree on a cryptosystem.
2. Alice and Bob agree on a key.

Symmetric-key Encryption Protocol

- Assuming that we have access to a symmetric-key cryptosystem (DES is an example), how do we use it?
- We have to describe a protocol that shows how each party uses the cryptosystem to solve a communication/security problem.

1. Alice and Bob agree on a cryptosystem.
2. Alice and Bob agree on a key.
3. Alice encrypts her plaintext, getting a ciphertext.

Symmetric-key Encryption Protocol

- Assuming that we have access to a symmetric-key cryptosystem (DES is an example), how do we use it?
- We have to describe a protocol that shows how each party uses the cryptosystem to solve a communication/security problem.

1. Alice and Bob agree on a cryptosystem.
2. Alice and Bob agree on a key.
3. Alice encrypts her plaintext, getting a ciphertext.
4. Alice sends the ciphertext to Bob.

Symmetric-key Encryption Protocol

- Assuming that we have access to a symmetric-key cryptosystem (DES is an example), how do we use it?
- We have to describe a protocol that shows how each party uses the cryptosystem to solve a communication/security problem.

1. Alice and Bob agree on a cryptosystem.
2. Alice and Bob agree on a key.
3. Alice encrypts her plaintext, getting a ciphertext.
4. Alice sends the ciphertext to Bob.
5. Bob decrypts the message using the same cryptosystem and key.

Symmetric Encryption Protocol. . .

Alice

plaintext

Symmetric Encryption Protocol. . .

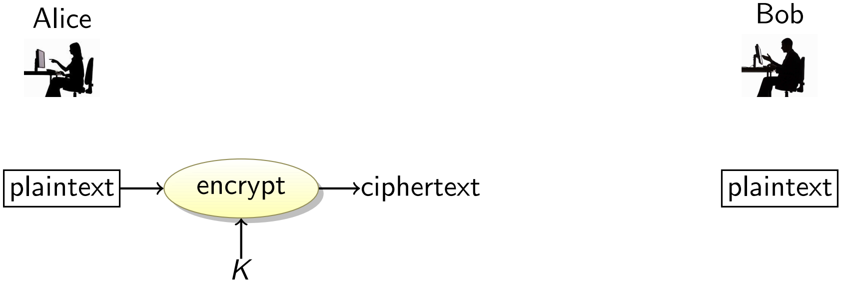
Alice

Bob

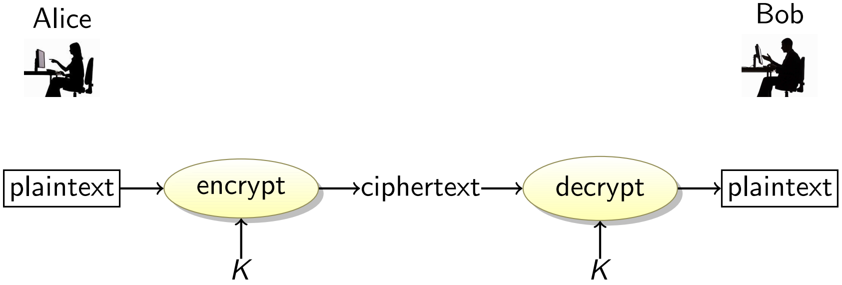
plaintext

plaintext

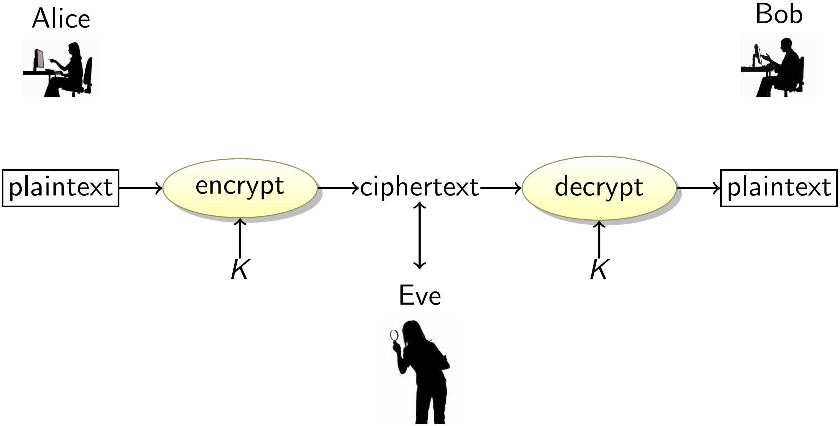
Symmetric Encryption Protocol. . .



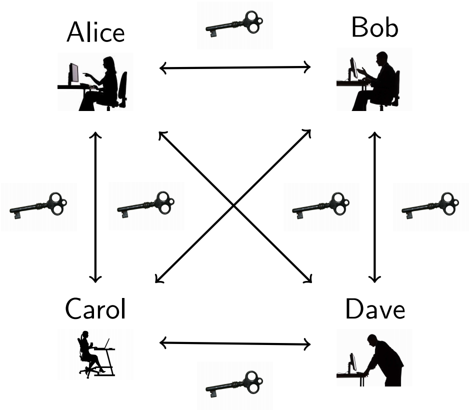
Symmetric Encryption Protocol. . .



Symmetric Encryption Protocol. . .



Symmetric Encryption: Key Distribution



- Advantages: Ciphers (DES,AES,...) are fast; keys are small.
- Disadvantages: $n\frac{n-1}{2}$ keys to communicate between n parties.

Symmetric Encryption Protocol – Attacks

- What can an attacker do?
- If Eve listens in on the communication between Alice and Bob she will get a sequence of ciphertext messages. She can use these to launch a ciphertext-only attack.
- Eve could also try to listen in to the first two parts of the protocol, where Alice and Bob decide on a key and cryptosystem to use.
- Eve could also sit in the middle, intercept Alice's messages, and substitute her own messages encrypted with the key she has discovered.

Public Key Protocol

- Key-management is the main problem with symmetric algorithms – Bob and Alice have to somehow agree on a key to use.
- In public key cryptosystems there are two keys, a public one used for encryption and and private one for decryption.

Public Key Protocol

- Key-management is the main problem with symmetric algorithms – Bob and Alice have to somehow agree on a key to use.
- In public key cryptosystems there are two keys, a public one used for encryption and and private one for decryption.

1. Alice and Bob agree on a public key cryptosystem.

Public Key Protocol

- Key-management is the main problem with symmetric algorithms – Bob and Alice have to somehow agree on a key to use.
- In public key cryptosystems there are two keys, a public one used for encryption and and private one for decryption.

1. Alice and Bob agree on a public key cryptosystem.
2. Bob sends Alice his public key, or Alice gets it from a public database.

Public Key Protocol

- Key-management is the main problem with symmetric algorithms – Bob and Alice have to somehow agree on a key to use.
- In public key cryptosystems there are two keys, a public one used for encryption and and private one for decryption.

1. Alice and Bob agree on a public key cryptosystem.
2. Bob sends Alice his public key, or Alice gets it from a public database.
3. Alice encrypts her plaintext using Bob's public key and sends it to Bob.

Public Key Protocol

- Key-management is the main problem with symmetric algorithms – Bob and Alice have to somehow agree on a key to use.
- In public key cryptosystems there are two keys, a public one used for encryption and and private one for decryption.

1. Alice and Bob agree on a public key cryptosystem.
2. Bob sends Alice his public key, or Alice gets it from a public database.
3. Alice encrypts her plaintext using Bob's public key and sends it to Bob.
4. Bob decrypts the message using his private key.

Notation

- Bob's public key: P_B
- Bob's secret key: S_B

$$\begin{aligned} E_{P_B}(M) &= C \\ D_{S_B}(C) &= M \\ D_{S_B}(E_{P_B}(M)) &= M \end{aligned}$$

Public Key Encryption Protocol…

Alice

plaintext

Public Key Encryption Protocol…

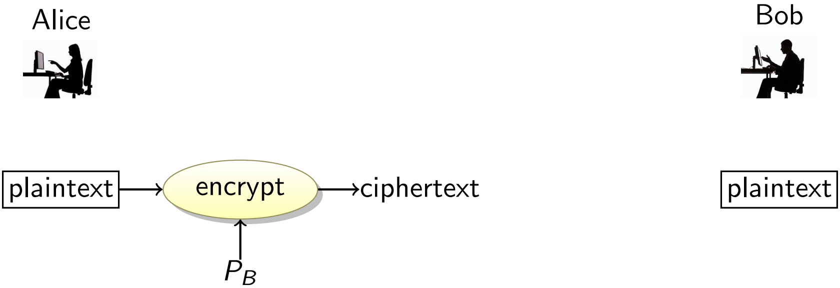
Alice

Bob

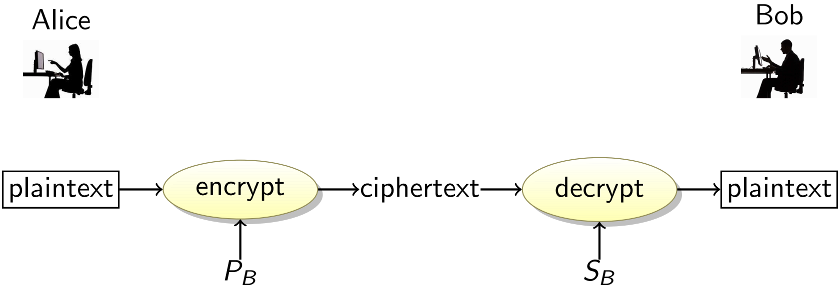
plaintext

plaintext

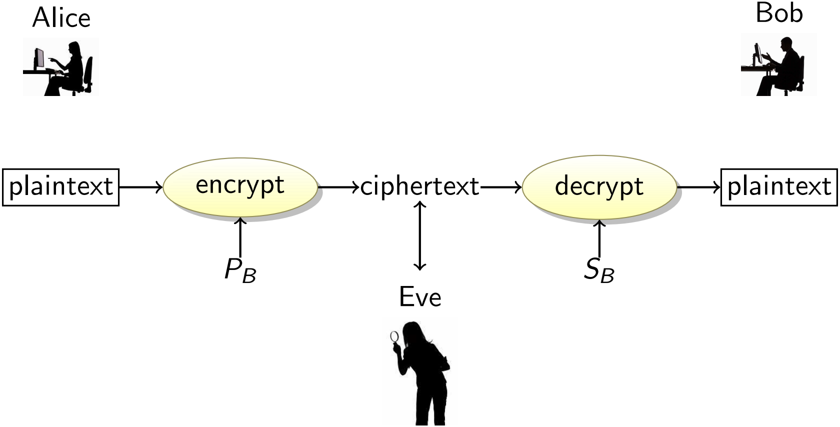
Public Key Encryption Protocol. . .



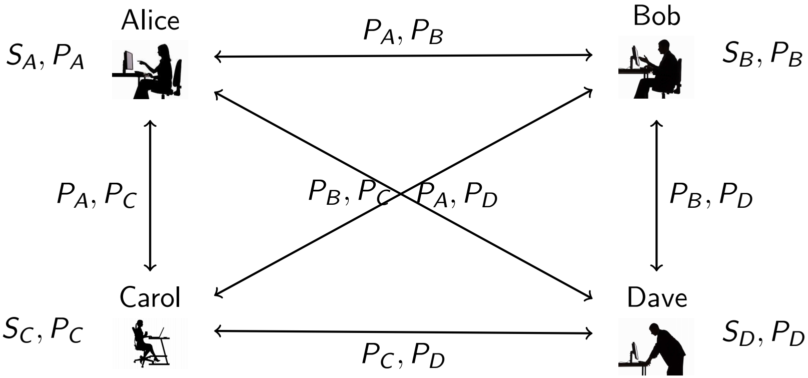
Public Key Encryption Protocol. . .



Public Key Encryption Protocol...



Public Key Encryption: Key Distribution



- Advantages: n key pairs to communicate between n parties.
- Disadvantages: Ciphers (RSA,...) are slow; keys are large

A Hybrid Protocol

- In practice, public key cryptosystems are not used to encrypt messages – they are simply too slow.
- Instead, public key cryptosystems are used to encrypt keys for symmetric cryptosystems. These are called session keys, and are discarded once the communication session is over.

A Hybrid Protocol

- In practice, public key cryptosystems are not used to encrypt messages – they are simply too slow.
- Instead, public key cryptosystems are used to encrypt keys for symmetric cryptosystems. These are called session keys, and are discarded once the communication session is over.

1. Bob sends Alice his public key.

A Hybrid Protocol

- In practice, public key cryptosystems are not used to encrypt messages – they are simply too slow.
- Instead, public key cryptosystems are used to encrypt keys for symmetric cryptosystems. These are called session keys, and are discarded once the communication session is over.

1. Bob sends Alice his public key.
2. Alice generates a session key K, encrypts it with Bob's public key, and sends it to Bob.

A Hybrid Protocol

- In practice, public key cryptosystems are not used to encrypt messages – they are simply too slow.
- Instead, public key cryptosystems are used to encrypt keys for symmetric cryptosystems. These are called session keys, and are discarded once the communication session is over.

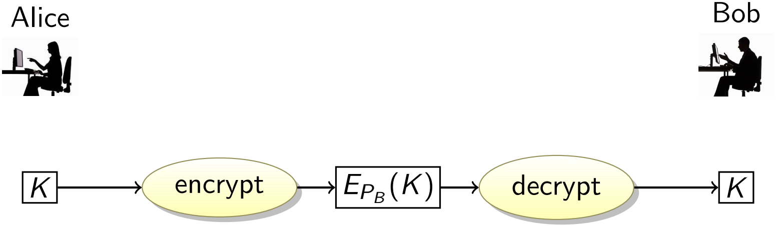
1. Bob sends Alice his public key.
2. Alice generates a session key K, encrypts it with Bob's public key, and sends it to Bob.
3. Bob decrypts the message using his private key to get the session key K.

A Hybrid Protocol

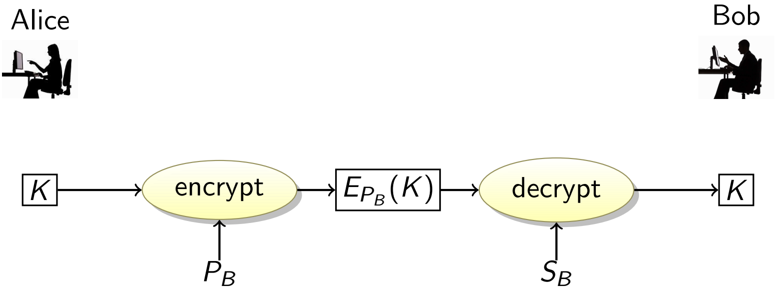
- In practice, public key cryptosystems are not used to encrypt messages – they are simply too slow.
- Instead, public key cryptosystems are used to encrypt keys for symmetric cryptosystems. These are called session keys, and are discarded once the communication session is over.

1. Bob sends Alice his public key.
2. Alice generates a session key K, encrypts it with Bob's public key, and sends it to Bob.
3. Bob decrypts the message using his private key to get the session key K.
4. Both Alice and Bob communicate by encrypting their messages using K.

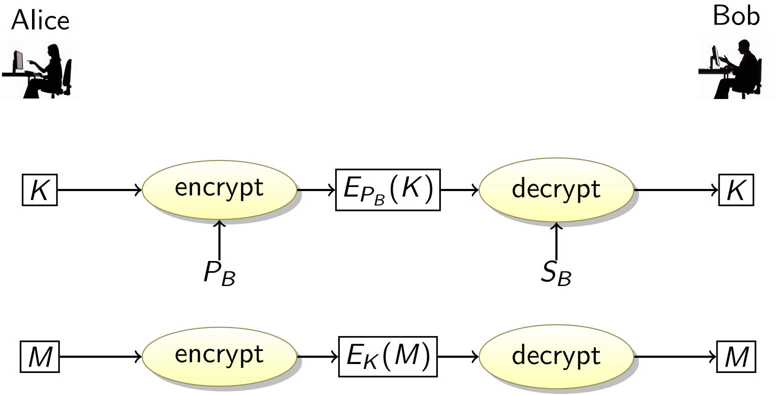
Hybrid Encryption Protocol. . .



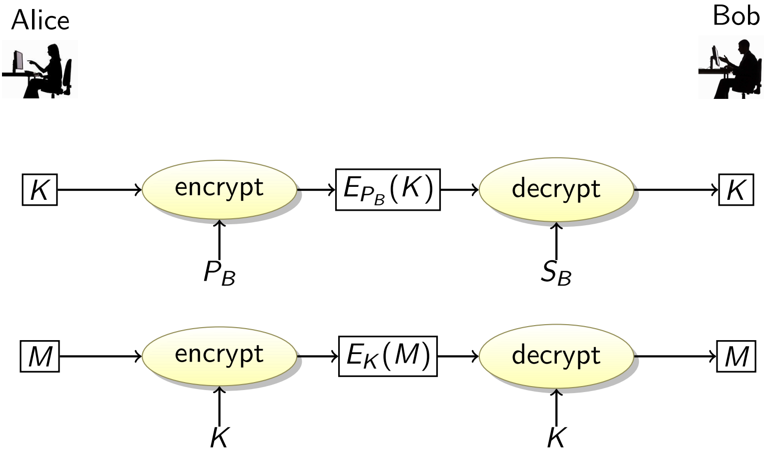
Hybrid Encryption Protocol. . .



Hybrid Encryption Protocol. . .



Hybrid Encryption Protocol...



Digital Signatures

- Often, Bob will want to make sure that the document he got from Alice in fact originated with her.
- In the non-digital world, Alice would sign the document. We can do the same with digital signatures.

Digital Signatures

- Often, Bob will want to make sure that the document he got from Alice in fact originated with her.
- In the non-digital world, Alice would sign the document. We can do the same with digital signatures.

1. Bob encrypts his document with his private key, thereby signing it.

Digital Signatures

- Often, Bob will want to make sure that the document he got from Alice in fact originated with her.
- In the non-digital world, Alice would sign the document. We can do the same with digital signatures.

1. Bob encrypts his document with his private key, thereby signing it.
2. Bob sends the signed document to Alice.

Digital Signatures

- Often, Bob will want to make sure that the document he got from Alice in fact originated with her.
- In the non-digital world, Alice would sign the document. We can do the same with digital signatures.

1. Bob encrypts his document with his private key, thereby signing it.
2. Bob sends the signed document to Alice.
3. Alice decrypts the document using Bob's public key, thereby verifying his signature.

Digital Signatures...

- This works because for many public key ciphers

$$
\begin{aligned}
D_{S_B}(E_{P_B}(M)) &= M \\
E_{P_B}(D_{S_B}(M)) &= M
\end{aligned}
$$

  i.e. we can reverse the encryption/decryption operations.

- That is, Bob can apply the decryption function to a message with his private key S_B, yielding the signature sig:

$$\mathrm{sig} \leftarrow D_{S_B}(M)$$

- Then, anyone else can apply the encryption function to sig to get the message back. Only Bob (who has his secret key) could have generated the signature:

$$E_{P_B}(\mathrm{sig}) = M$$

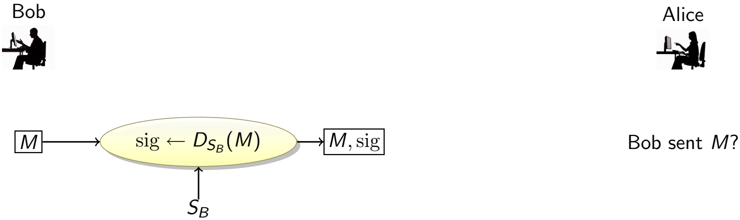
Digital Signatures. . .

- Disadvantages: the signature is as long as the message; susceptible to MITM attack.

Digital Signatures. . .

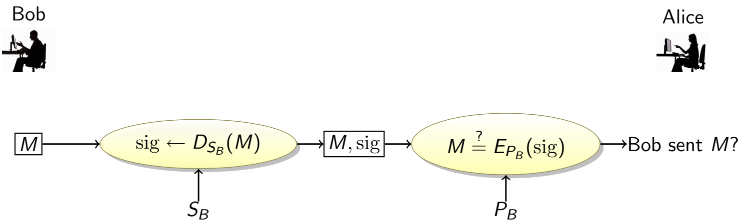
- Disadvantages: the signature is as long as the message; susceptible to MITM attack.

Digital Signatures...



- Disadvantages: the signature is as long as the message; susceptible to MITM attack.

Digital Signatures. . .



- Disadvantages: the signature is as long as the message; susceptible to MITM attack.

Digital Signatures: Man-In-The-Middle attack

- Eve can launch a MITM attack:

Digital Signatures: Man-In-The-Middle attack

- Eve can launch a MITM attack:
  1. Intercept Alice's message M, S to Bob.

Digital Signatures: Man-In-The-Middle attack

- Eve can launch a MITM attack:
  1. Intercept Alice's message M, S to Bob.
  2. Create a new signature string S'.

Digital Signatures: Man-In-The-Middle attack

- Eve can launch a MITM attack:
  1. Intercept Alice's message M, S to Bob.
  2. Create a new signature string S'.
  3. Create a message M' by encrypting S' with Bob's public key:

$$M' \leftarrow E_{P_B}(S').$$

Digital Signatures: Man-In-The-Middle attack

- Eve can launch a MITM attack:
  1. Intercept Alice's message M, S to Bob.
  2. Create a new signature string S'.
  3. Create a message M' by encrypting S' with Bob's public key:

$$M' \leftarrow E_{P_B}(S').$$

  4. Send M', S' to Alice.

Digital Signatures: Man-In-The-Middle attack

- Eve can launch a MITM attack:
  1. Intercept Alice's message M, S to Bob.
  2. Create a new signature string S'.
  3. Create a message M' by encrypting S' with Bob's public key:

$$M' \leftarrow E_{P_B}(S').$$

  4. Send M', S' to Alice.
- Alice:

Digital Signatures: Man-In-The-Middle attack

- Eve can launch a MITM attack:
  1. Intercept Alice's message M, S to Bob.
  2. Create a new signature string S'.
  3. Create a message M' by encrypting S' with Bob's public key:

$$M' \leftarrow E_{P_B}(S').$$

  4. Send M', S' to Alice.
- Alice:
  1. Alice receives M', S' from Eve (thinking it's from Bob).

Digital Signatures: Man-In-The-Middle attack

- Eve can launch a MITM attack:
  1. Intercept Alice's message M, S to Bob.
  2. Create a new signature string S'.
  3. Create a message M' by encrypting S' with Bob's public key:

$$M' \leftarrow E_{P_B}(S').$$

  4. Send M', S' to Alice.
- Alice:
  1. Alice receives M', S' from Eve (thinking it's from Bob).
  2. She encrypts S' with Bob's public key: $v \leftarrow E_{P_B}(S')$.

Digital Signatures: Man-In-The-Middle attack

- Eve can launch a MITM attack:
  1. Intercept Alice's message M, S to Bob.
  2. Create a new signature string S'.
  3. Create a message M' by encrypting S' with Bob's public key:

$$M' \leftarrow E_{P_B}(S').$$

  4. Send M', S' to Alice.
- Alice:
  1. Alice receives M', S' from Eve (thinking it's from Bob).
  2. She encrypts S' with Bob's public key: $v \leftarrow E_{P_B}(S')$.
  3. She verifies: $v \stackrel{?}{=} M'$.

Digital Signatures: Man-In-The-Middle attack

- Eve can launch a MITM attack:
  1. Intercept Alice's message M, S to Bob.
  2. Create a new signature string S'.
  3. Create a message M' by encrypting S' with Bob's public key:

$$M' \leftarrow E_{P_B}(S').$$

  4. Send M', S' to Alice.
- Alice:
  1. Alice receives M', S' from Eve (thinking it's from Bob).
  2. She encrypts S' with Bob's public key: $v \leftarrow E_{P_B}(S')$.
  3. She verifies: $v \stackrel{?}{=} M'$.
- Note: Eve can't choose her own message. This attack only makes sense if any bitstring forms an OK message, for example if M is a session key.

Digital Signatures: Man-In-The-Middle attack. . .

Bob

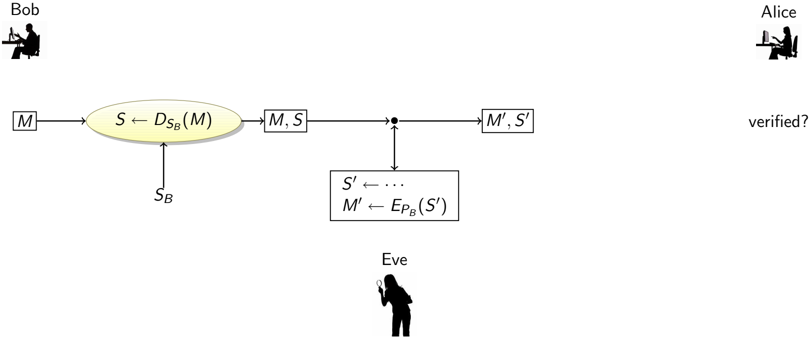
Alice

$\boxed{M}$

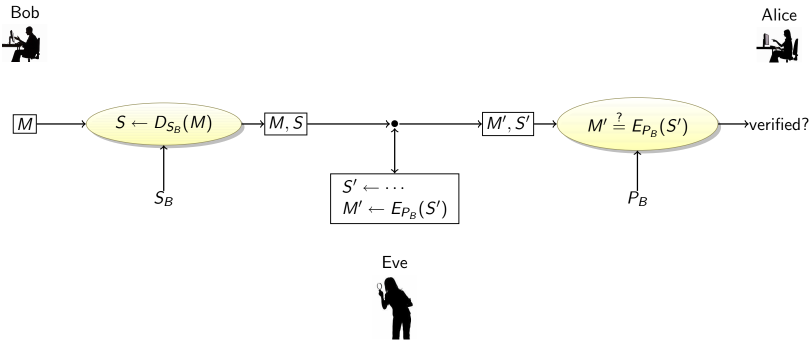
verified?

Digital Signatures: Man-In-The-Middle attack. . .

Digital Signatures: Man-In-The-Middle attack. . .



Digital Signatures: Man-In-The-Middle attack. . .



Cryptographic Hash Functions

- Public key algorithms are too slow to sign large documents. A better protocol is to use a one way hash function also known as a cryptographic hash function (CHF).
- CHFs are checksums or compression functions: they take an arbitrary block of data and generate a unique, short, fixed-size, bitstring.

```
> echo "hello" | sha1sum
f572d396fae9206628714fb2ce00f72e94f2258f  -
> echo "hella" | sha1sum
1519ca327399f9d699afb0f8a3b7e1ea9d1edd0c  -
> echo "can't believe it's not butter!"|sha1sum
34e780e19b07b003b7cf1babba8ef7399b7f81dd  -
```

Signature Protocol

1. Bob computes a one-way hash of his document.

$$\begin{aligned} \text{hash} &\leftarrow h(M) \\ \text{sig} &\leftarrow E_{S_B}(\text{hash}) \\ D_{P_B}(\text{sig}) &\stackrel{?}{=} h(M) \end{aligned}$$

Signature Protocol

1. Bob computes a one-way hash of his document.
2. Bob encrypts the hash with his private key, thereby signing it.

$$\begin{aligned} \text{hash} &\leftarrow h(M) \\ \text{sig} &\leftarrow E_{S_B}(\text{hash}) \\ D_{P_B}(\text{sig}) &\stackrel{?}{=} h(M) \end{aligned}$$

Signature Protocol

1. Bob computes a one-way hash of his document.
2. Bob encrypts the hash with his private key, thereby signing it.
3. Bob sends the encrypted hash and the document to Alice.

$$\begin{aligned} \text{hash} &\leftarrow h(M) \\ \text{sig} &\leftarrow E_{S_B}(\text{hash}) \\ D_{P_B}(\text{sig}) &\stackrel{?}{=} h(M) \end{aligned}$$

Signature Protocol

1. Bob computes a one-way hash of his document.
2. Bob encrypts the hash with his private key, thereby signing it.
3. Bob sends the encrypted hash and the document to Alice.
4. Alice decrypts the hash Bob sent him, and compares it against a hash she computes herself of the document. If they are the same, the signature is valid.

$$
\begin{aligned}
\text{hash} &\leftarrow h(M) \\
\text{sig} &\leftarrow E_{S_B}(\text{hash}) \\
D_{P_B}(\text{sig}) &\stackrel{?}{=} h(M)
\end{aligned}
$$

Signature Protocol. . .

Bob

Alice

$\boxed{M}$

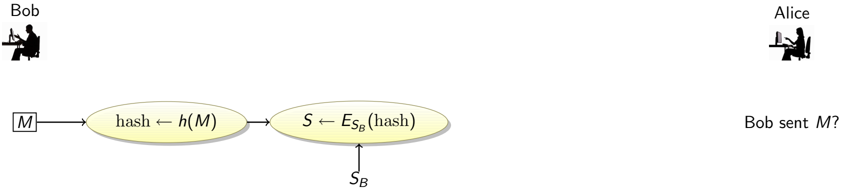
Bob sent M?

- Advantage: the signature is short; defends against MITM attack.

Signature Protocol...

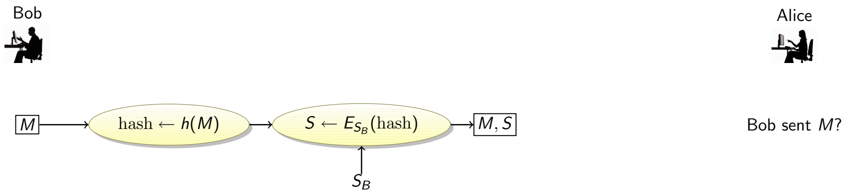
- Advantage: the signature is short; defends against MITM attack.

Signature Protocol. . .



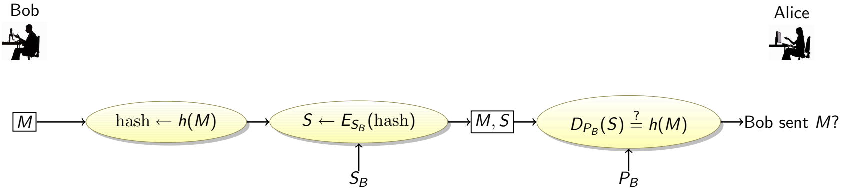
- Advantage: the signature is short; defends against MITM attack.

Signature Protocol. . .



- Advantage: the signature is short; defends against MITM attack.

Signature Protocol...



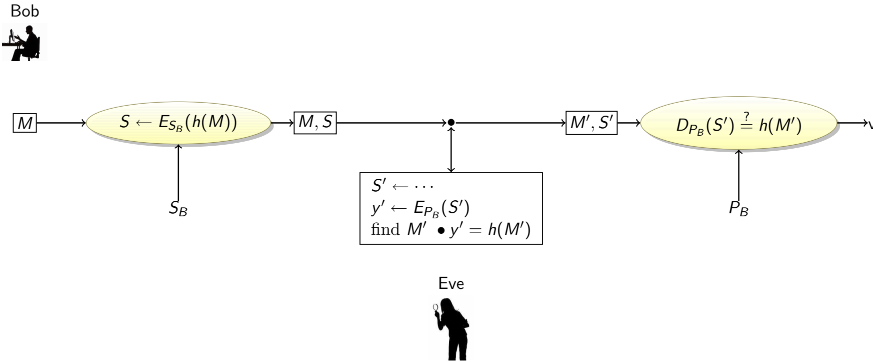
- Advantage: the signature is short; defends against MITM attack.

Cryptographic Hash Functions...

- CHFs are easy to compute, but hard to invert.
- I.e.
  - given message M, it's easy to compute $y \leftarrow h(M)$;
  - given a value y it's hard to compute an M such that $y = h(M)$.

  This is what we mean by CHFs being one-way.
- Using the one-way property, we can defend against the Man-In-The-Middle attack.

Cryptographic Hash Functions: MITM attack



- Eve has to find a message M' that hashes to y', the result of encrypting the forged signature S'.
- This is infeasible since h is one-way.

Cryptographic Hash Functions. . .

- CHFs also have the property to be collision resistant.
- I.e. given M it's hard to find a different message M' such that $h(M) = h(M')$.
- This makes Eve's job even harder: given M, S from Bob she has to find a different message M' that has the same signature S.

Cryptographic Hash Functions: tripwire

1. When initializing a system, store hashes of all important files in secure storage.
2. Detect tampering by re-computing the hashes and comparing against the original values.

```
> sha1sum /etc/passwd > okfiles
> sha1sum --check okfiles
/etc/passwd: OK
```

- http://sourceforge.net/projects/tripwire.

Message Authentication Codes (MAC)

- MACs are used to ensure the integrity of messages sent over insecure channels.
- We assume that Alice and Bob have a shared secret symmetric key K.
- Alice wants to send message M to Bob:

Message Authentication Codes (MAC)

- MACs are used to ensure the integrity of messages sent over insecure channels.
- We assume that Alice and Bob have a shared secret symmetric key K.
- Alice wants to send message M to Bob:
  1. $A \leftarrow h(K||M)$

Message Authentication Codes (MAC)

- MACs are used to ensure the integrity of messages sent over insecure channels.
- We assume that Alice and Bob have a shared secret symmetric key K.
- Alice wants to send message M to Bob:
  1. $A \leftarrow h(K||M)$
  2. Send M, A to Bob.

Message Authentication Codes (MAC)

- MACs are used to ensure the integrity of messages sent over insecure channels.
- We assume that Alice and Bob have a shared secret symmetric key K.
- Alice wants to send message M to Bob:
  1. $A \leftarrow h(K||M)$
  2. Send M, A to Bob.
- Bob:

Message Authentication Codes (MAC)

- MACs are used to ensure the integrity of messages sent over insecure channels.
- We assume that Alice and Bob have a shared secret symmetric key K.
- Alice wants to send message M to Bob:
  1. $A \leftarrow h(K||M)$
  2. Send M, A to Bob.
- Bob:
  1. Receive M', A' from Alice

Message Authentication Codes (MAC)

- MACs are used to ensure the integrity of messages sent over insecure channels.
- We assume that Alice and Bob have a shared secret symmetric key K.
- Alice wants to send message M to Bob:
  1. $A \leftarrow h(K||M)$
  2. Send M, A to Bob.
- Bob:
  1. Receive M', A' from Alice
  2. $A'' \leftarrow h(K||M')$

Message Authentication Codes (MAC)

- MACs are used to ensure the integrity of messages sent over insecure channels.
- We assume that Alice and Bob have a shared secret symmetric key K.
- Alice wants to send message M to Bob:
  1. $A \leftarrow h(K||M)$
  2. Send M, A to Bob.
- Bob:
  1. Receive M', A' from Alice
  2. $A'' \leftarrow h(K||M')$
  3. Verify: $A'' \stackrel{?}{=} A'$.

Message Authentication Codes (MAC)

- MACs are used to ensure the integrity of messages sent over insecure channels.
- We assume that Alice and Bob have a shared secret symmetric key K.
- Alice wants to send message M to Bob:
  1. $A \leftarrow h(K||M)$
  2. Send M, A to Bob.
- Bob:
  1. Receive M', A' from Alice
  2. $A'' \leftarrow h(K||M')$
  3. Verify: $A'' \stackrel{?}{=} A'$.
- $||$ means concatenation.

Message Authentication Codes (MAC)

- MACs are used to ensure the integrity of messages sent over insecure channels.
- We assume that Alice and Bob have a shared secret symmetric key K.
- Alice wants to send message M to Bob:
  1. $A \leftarrow h(K||M)$
  2. Send M, A to Bob.
- Bob:
  1. Receive M', A' from Alice
  2. $A'' \leftarrow h(K||M')$
  3. Verify: $A'' \stackrel{?}{=} A'$.
- $||$ means concatenation.
- The one-way nature of h makes it infeasible for Eve to extract K from $A = h(K||M)$, and without K she can't forge a new message with a correct MAC.

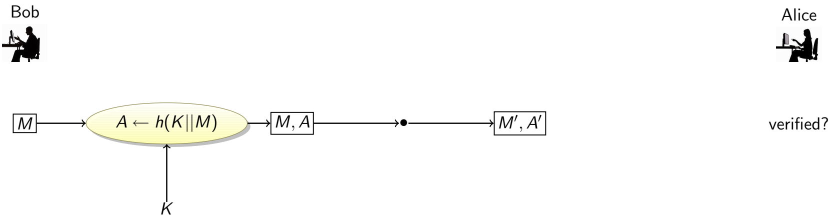
MACS: MITM attack

- Eve has to recover K from A — infeasible since h is one-way.

MACS: MITM attack

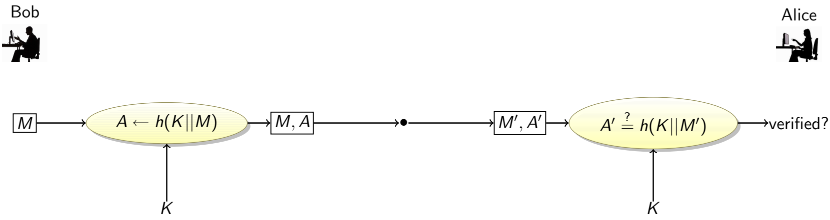
- Eve has to recover K from A — infeasible since h is one-way.

MACS: MITM attack



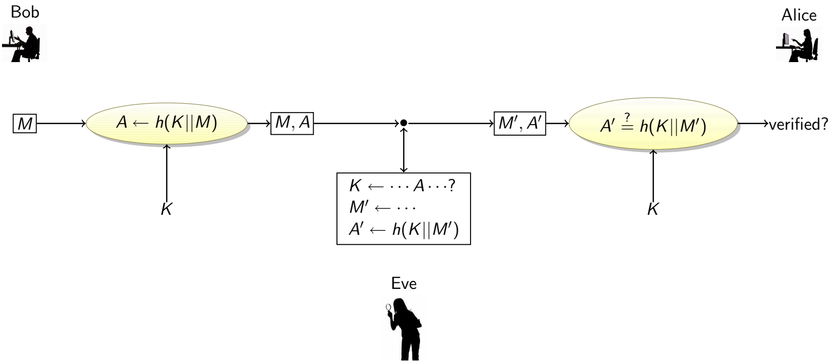
- Eve has to recover K from A — infeasible since h is one-way.

MACS: MITM attack



- Eve has to recover K from A — infeasible since h is one-way.

MACS: MITM attack



- Eve has to recover K from A — infeasible since h is one-way.

MACs vs. Digital Signatures

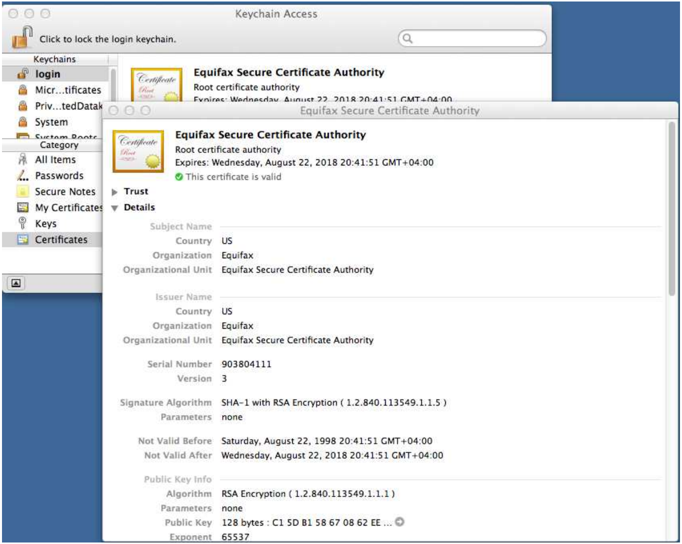
- MACs use shared secret symmetric keys, digital signatures use public keys.
- Digital signatures also protect against non-repudiation, i.e., Alice can't deny that she's the one who sent a particular message.
- Digital signatures are transferable, i.e. if Bob receives M, S from Alice, he can send on that message/signature pair to Charles who can verify that it's Alice who sent that message.

Digital Certificates

- How does Alice know that P_B is actually Bob's public key? What if there are many Bobs?
- A Certificate Authority (CA) is a trusted third party (TTT) who issues a certificate stating that

  > *The Bob who lives on Desolation Row and has phone number (555) 867-5309 and the email address* `bob@gmail.com` *has the public key P_B. This certificate is valid until June 11, 2012.*

- The CA has to digitally sign (with their private key S_{CA}) this certificate so that we know that it's real.



- List of certificates in the Chrome browser.

Digital Certificates: X.509

- Certificate
  - Version
  - Serial Number
  - Algorithm ID
  - Issuer
  - Validity: [Not Before..Not After]
  - Subject
  - Subject Public Key Info
    - Public Key Algorithm
    - Subject Public Key
  - Issuer Unique Identifier (optional)
  - Subject Unique Identifier (optional)
- Certificate Signature Algorithm
- Certificate Signature

Digital Certificates: chase.com

- In Safari:
  1. Go to chase.com
  2. click on the padlock
  3. command-drag the certificate, creating the file chaseonline.chase.com.txt.
  4. Convert to unix newlines:

```
> tr '\r' '\n' \
   < chaseonline.chase.com.txt \
   > chase.txt
```

Digital Certificates: chase.com I

```
chaseonline.chase.com

Subject Name
    Country US
    State/Province Ohio
    Locality Columbus
    Organization JPMorgan Chase
    Organizational Unit cig1w156
    Common Name chaseonline.chase.com

Issuer Name
    Country US
    Organization VeriSign, Inc.
    Organizational Unit VeriSign Trust Network
```

Digital Certificates: chase.com II

```
    Common Name VeriSign Class 3 International ...

Serial Number 11 D4 0D 20 EE 53 E1 91 19 38 4C ...
Version 3
Signature Algorithm SHA-1 with RSA Encryption ...
    Parameters none

Not Valid Before Wednesday, April 27, 2011 17:00:00 MST
Not Valid After Friday, May 18, 2012 16:59:59 MST

Public Key Info
    Algorithm RSA Encryption
    Parameters none
    Public Key 256 bytes : A7 15 F2 F5 BD AB FE D0 ...
    Exponent 65537
```

Digital Certificates: `chase.com` III

```
    Key Size 2048 bits
    Key Usage Encrypt, Verify, Wrap, Derive

Signature 256 bytes : 76 9B D8 C5 77 1E CB 01 ...

Extension Key Usage ( 2.5.29.15 )
    Critical NO
    Usage Digital Signature, Key Encipherment

Fingerprints
    SHA1 DF BF D3 7A 93 15 E9 ED CD 44 D8 ...
    MD5        8B 60 1E B0 5F 69 59 52 80 E2 72 ...
```

Digital Certificates: Securely connecting to `chase.com`

- To do online banking with `chase.com`, Alice wants to ensure that
  - the web site is who it claims to be,
  - no one is eavesdropping on her interaction with the web.

Digital Certificates: Securely connecting to chase.com

1. Alice browses to https://chase.com

Digital Certificates: Securely connecting to `chase.com`

1. Alice browses to `https://chase.com`
2. The browser asks `chase.com` to identify itself.

Digital Certificates: Securely connecting to `chase.com`

1. Alice browses to `https://chase.com`
2. The browser asks `chase.com` to identify itself.
3. `chase.com` returns its certificate to the browser.

Digital Certificates: Securely connecting to `chase.com`

1. Alice browses to `https://chase.com`
2. The browser asks `chase.com` to identify itself.
3. `chase.com` returns its certificate to the browser.
4. Extract $\mathrm{CA} \leftarrow$ from the certificate. The certificate is signed with S_{CA}.

Digital Certificates: Securely connecting to `chase.com`

1. Alice browses to `https://chase.com`
2. The browser asks `chase.com` to identify itself.
3. `chase.com` returns its certificate to the browser.
4. Extract $\mathrm{CA} \leftarrow$ from the certificate. The certificate is signed with S_{CA}.
5. The browser checks if it trusts the certificate:

Digital Certificates: Securely connecting to `chase.com`

1. Alice browses to `https://chase.com`
2. The browser asks `chase.com` to identify itself.
3. `chase.com` returns its certificate to the browser.
4. Extract $\mathrm{CA} \leftarrow$ from the certificate. The certificate is signed with S_{CA}.
5. The browser checks if it trusts the certificate:
   - Do we trust the CA? The browser has public keys P_{CA} of trusted CAs pre-installed.

Digital Certificates: Securely connecting to `chase.com`

1. Alice browses to `https://chase.com`
2. The browser asks `chase.com` to identify itself.
3. `chase.com` returns its certificate to the browser.
4. Extract $\mathrm{CA} \leftarrow$ from the certificate. The certificate is signed with S_{CA}.
5. The browser checks if it trusts the certificate:
   - Do we trust the CA? The browser has public keys P_{CA} of trusted CAs pre-installed.
   - Has the certificate expired?

Digital Certificates: Securely connecting to `chase.com`

1. Alice browses to `https://chase.com`
2. The browser asks `chase.com` to identify itself.
3. `chase.com` returns its certificate to the browser.
4. Extract $\mathrm{CA} \leftarrow$ from the certificate. The certificate is signed with S_{CA}.
5. The browser checks if it trusts the certificate:
   - Do we trust the CA? The browser has public keys P_{CA} of trusted CAs pre-installed.
   - Has the certificate expired?
6. The browser generates a session-key K;

Digital Certificates: Securely connecting to `chase.com`

1. Alice browses to `https://chase.com`
2. The browser asks `chase.com` to identify itself.
3. `chase.com` returns its certificate to the browser.
4. Extract $\mathrm{CA} \leftarrow$ from the certificate. The certificate is signed with S_{CA}.
5. The browser checks if it trusts the certificate:
   - Do we trust the CA? The browser has public keys P_{CA} of trusted CAs pre-installed.
   - Has the certificate expired?
6. The browser generates a session-key K;
7. Extract $P_{\texttt{chase.com}} \leftarrow$ from the certificate.

Digital Certificates: Securely connecting to chase.com

1. Alice browses to https://chase.com
2. The browser asks chase.com to identify itself.
3. chase.com returns its certificate to the browser.
4. Extract $\mathrm{CA} \leftarrow$ from the certificate. The certificate is signed with S_{CA}.
5. The browser checks if it trusts the certificate:
   - Do we trust the CA? The browser has public keys P_{CA} of trusted CAs pre-installed.
   - Has the certificate expired?
6. The browser generates a session-key K;
7. Extract $P_{\texttt{chase.com}} \leftarrow$ from the certificate.
8. The browser encrypts K with $P_{\texttt{chase.com}}$

Digital Certificates: Securely connecting to chase.com

1. Alice browses to https://chase.com
2. The browser asks chase.com to identify itself.
3. chase.com returns its certificate to the browser.
4. Extract $\mathrm{CA} \leftarrow$ from the certificate. The certificate is signed with S_{CA}.
5. The browser checks if it trusts the certificate:
   - Do we trust the CA? The browser has public keys P_{CA} of trusted CAs pre-installed.
   - Has the certificate expired?
6. The browser generates a session-key K;
7. Extract $P_{\texttt{chase.com}} \leftarrow$ from the certificate.
8. The browser encrypts K with $P_{\texttt{chase.com}}$
9. The browser sends $P_{\texttt{chase.com}}(K)$ to chase.com.

In-Class Exercise I — Goodrich & Tamassia R-1.12

- What are the strengths and weaknesses of symmetric-key encryption and public-key encryption?

In-Class Exercise II— Goodrich & Tamassia C-1.10

- Alice and Bob are communicating using public key cryptography.
- Bob is Alice's bookie.
- Bob send messages of the form E_{P_A}(3rd race @ saratoga?).
- Alice responds with a message of the form E_{P_B}(\$100 on Golden Mane).
- Eve knows P_A and P_B, the form of the messages, that Alice only bets in multiples of \$100 and never more than \$1000, and all the races and all the horses at all the race tracks (easy to get via a web search).
- How can Eve learn what Alice is betting?

In-Class Exercise III — Goodrich & Tamassia C-1.11

- Can you think of a way to prevent Eve in the previous exercise from learning the contents of the communication?

In-Class Exercise IV — Goodrich & Tamassia C-1.12

- Alice is full of good ideas for new startups that she wants to send to Bob.
- She wants to make sure that Charles can't take credit for her ideas.
- How can she achieve this using public-key cryptography?

In-Class Exercise V — Goodrich & Tamassia C-1.13

- Alice is full of good ideas for new startups that she wants to send to Bob.
- She wants to make sure that Charles can't take credit for her ideas.
- How can she achieve this using symmetric-key cryptography?

In-Class Exercise VI — Goodrich & Tamassia C-1.14

- Alice and Daisy are both full of good ideas for new startups that they want to send to Bob.
- Each wants to make sure that the other cannot take credit for their ideas.
- Alice, Daisy, and Bob therefore share a secret key K.
- Along with each message M, they send $d = h(K||M)$.
- Can Bob verify who sent him a particular idea?

In-Class Exercise VII — Goodrich & Tamassia C-1.17

- Alice and Bob want to verify they have the same secret n-bit key K. They engage in the following protocol:
  1. Alice generates a random n-bit value R.
  2. Alice sends $X \leftarrow K_A \oplus R$ to Bob ($\oplus$ = exclusive-or).
  3. Bob sends $Y \leftarrow K_B \oplus X$ to Alice.
  4. Alice compares R and Y. If $R = Y$, she concludes that $K_A = K_B$.
- How can Eve recover the keys?

Outline

Readings

- Chapter 1 in *Introduction to Computer Security*, by Goodrich and Tamassia.

Acknowledgments

Material and exercises have also been collected from these sources:

1. Bishop, *Introduction to Computer Security*.