CSc 466/566

Computer Security

19 : Cryptography — Protocols II

Version: 2019/11/04 12:48:54

Department of Computer Science
University of Arizona

collberg@gmail.com

Christian Collberg

Outline

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.

Protocol

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:

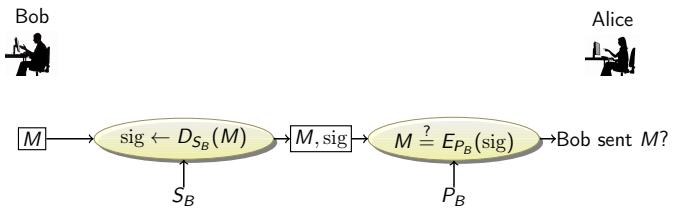
$$\mathrm{sig} \leftarrow D_{S_B}(M)$$

Digital Signatures. . .

- 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 — Protocol



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

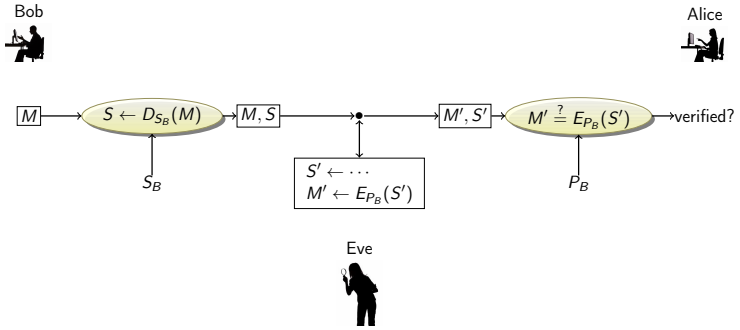
MITM attack

- Eve launches a Man-In-The-Middle 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: $v \stackrel{?}{=} M'$.

MITM attack...



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.

Exercise — 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?

Outline

Cryptographic Hash Functions

- Public key algorithms are too slow to sign large documents.
- A better protocol is to use a one way hash function (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.

Avalanche Effect...

```
> echo hello | openssl dgst -sha1
(stdin)= f572d396fae9206628714fb2ce00f72e94f2258f
> echo hella | openssl dgst -sha1
(stdin)= 1519ca327399f9d699afb0f8a3b7e1ea9d1edd0c
> echo helo | openssl dgst -sha1
(stdin)= 1904c437b1d849a7bdfb4a78b4e56fec785dbcdd
```

- Avalanche Effect: when an input is changed slightly the output changes significantly. https://en.wikipedia.org/wiki/Avalanche_effect

Common Hash Functions

- MD5: 128-bit hash
- SHA-1: 160-bit hash
- SHA-256: 256-bit hash

```
> openssl dgst -sha1 /etc/passwd
SHA1(/etc/passwd)= c0c71bac843a3ec7233e99e123888beb6da8fbcf
> openssl dgst -md5 /etc/passwd
MD5(/etc/passwd)= 5e7f80888f3d491c4963881364048c24
> openssl dgst -sha256 /etc/passwd
SHA256(/etc/passwd)= 39c487734fed185cf16217552ed8b\
   451525c240e13d41001b3782b46fdcf4708
```


Pre-image resistance

- Given message M, it's easy to compute $y \leftarrow h(M)$:

```
> echo helo | openssl dgst -sha1
(stdin)= 1904c437b1d849a7bdfb4a78b4e56fec785dbcdd
```

- But, given a value y it's hard to find an M such that $y = h(M)$:

```
596c3e3c9f5f407b6df7062aefdca9462b707dc6 = sha1(M)
```

- (The answer is $M =$ "bella").

Pre-image resistance...

- CHFs are easy to compute, but hard to invert.
- This is what we mean by CHFs being one-way.
- Also known as pre-image resistance.

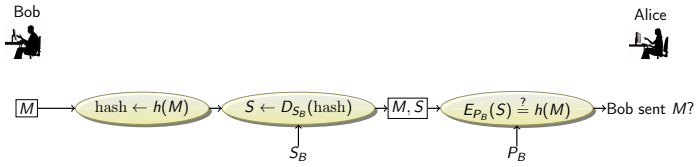
Signature Protocol

1. Bob:
   1. computes a one-way hash of his document.
      $$\text{hash} \leftarrow h(M)$$
   2. decrypts the hash with his private key, thereby signing it.
      $$\text{sig} \leftarrow D_{S_B}(\text{hash})$$
   3. sends the encrypted hash and the document to Alice.

Signature Protocol. . .

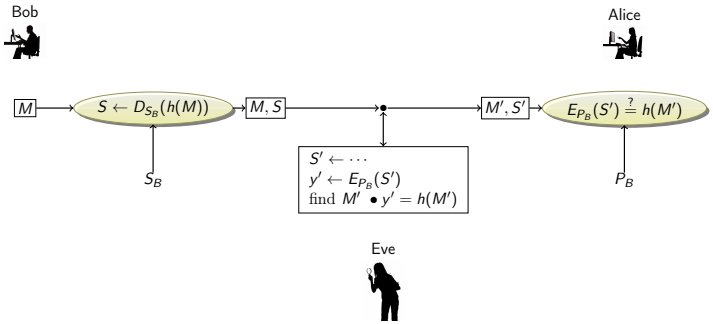
2. Alice:
   1. encrypts the hash Bob sent her using Bob's public key.
   2. compares it against a hash she computes herself of the document.
      $$E_{P_B}(\text{sig}) \stackrel{?}{=} h(M)$$
      If they are the same, the signature is valid.

Signature Protocol. . .



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

Attempted 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.

Collision Resistance

- 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.

Collision Resistance — Example

- Given that

```
> echo helo | openssl dgst -sha1
(stdin)= 1904c437b1d849a7bdfb4a78b4e56fec785dbcdd
```

  find another message M' such that

```
> echo M' | openssl dgst -sha1
(stdin)= 1904c437b1d849a7bdfb4a78b4e56fec785dbcdd
```

- For a good (one-way) hash function, this is infeasible.

SHA1 Attack

- Recently, SHA-1 hash collisions were found:

```
> openssl dgst -sha1 shattered-1.pdf shattered-2.pdf
SHA1(shattered-1.pdf)= 38762
    cf7f55934b34d179ae6a4c80cadccbb7f0a
SHA1(shattered-2.pdf)= 38762
    cf7f55934b34d179ae6a4c80cadccbb7f0a
```

- This attack required over 9,223,372,036,854,775,808 SHA1 computations: 6,500 years of single-CPU computations and 110 years of single-GPU computations.
- It cost $110,000 to carry out on Amazon's cloud.
- Source: http://thehackernews.com/2017/02/sha1-collision-attack.html

Outline

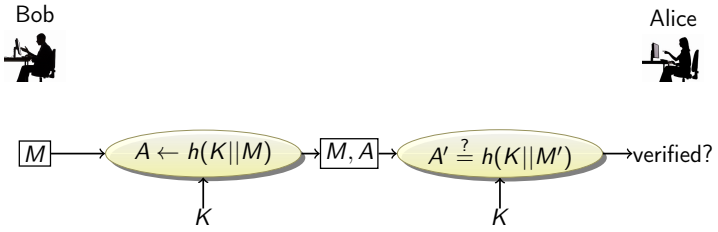
Message Authentication Codes (MAC)

- MACs are used to ensure the integrity of messages sent over insecure channels.
- We assume Alice and Bob have a shared secret symmetric key K.
- $h()$ is a cryptographic hash function.
- $||$ means concatenation.

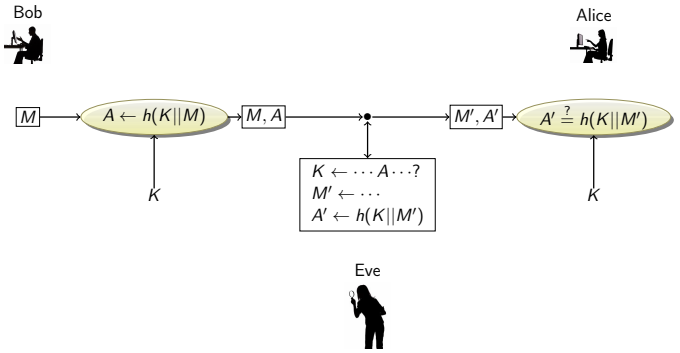
Protocol

- 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: $A'' \stackrel{?}{=} A'$.
- 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.

Protocol...



Attempted MITM attack



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

Openssl HMACs

```
> cat key.bin
1234567890
> cat message.bin
hello world
> cat message.bin | \
   openssl dgst -sha256 -hmac "`cat key.bin`" \
      -binary > mac.bin
```


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.

Exercise — 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?

Exercise — 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?

Exercise — Midterm 2012

Alice and Bob are involved in a torrid affair. They exchange short messages of the form

```
<hour:minute>,<name-of-hotel>
```

describing the time and place where they are going to meet. In order to avoid being detected by Eve, Bob's cryptographer wife, they encrypt the messages using 4,048 bit RSA keys.

1. How can Eve discover where and when she can catch the two love-birds in the act?

Exercise — Midterm 2012...

3. Describe two ways in which Alice and Bob can modify their protocol to avoid being discovered!

Exercise — Midterm 2012...

3. Alice and Bob are communicating over an insecure channel and are worried that their messages will be tampered with, resulting in them showing up in the wrong hotel at the wrong time. Describe, in detail, how they can use *symmetric key cryptography* to avoid this.

Outline

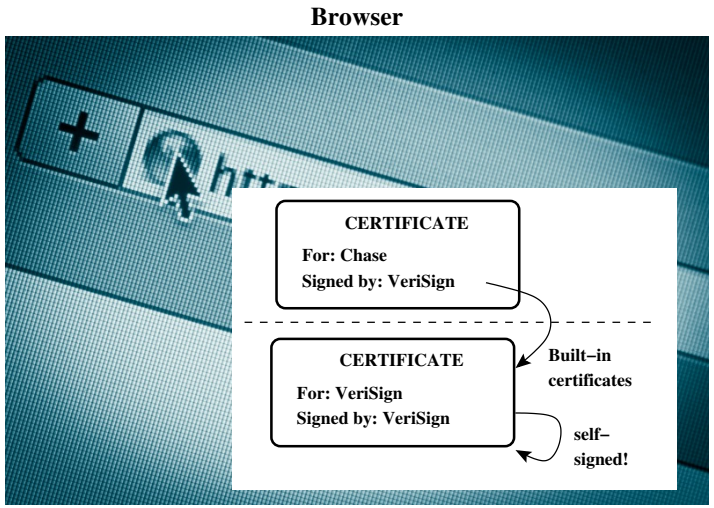
Digital Certificates

- 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 (TTP) who issues a certificate stating that

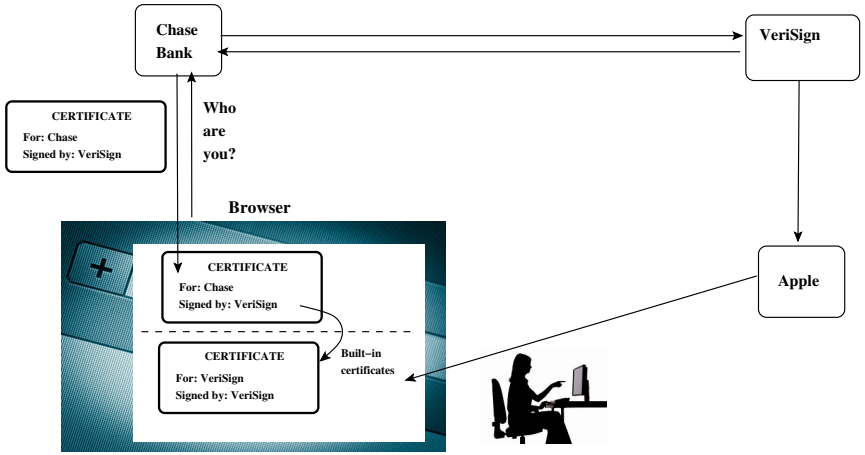
  The Bob who lives on Desolation Row and has phone number (555) 867-5309 and the email address bob@gmail.com *has the public key P_B. This certificate is valid until June 11, 2012.*

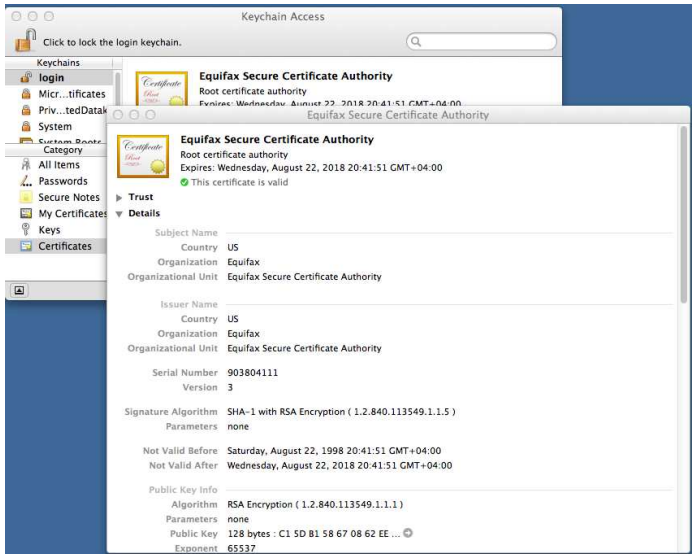
- The CA has to digitally sign (with their private key S_{CA}) this certificate so that we know that it's real.

Digital Certificates in the Browser



Issuing Certificates





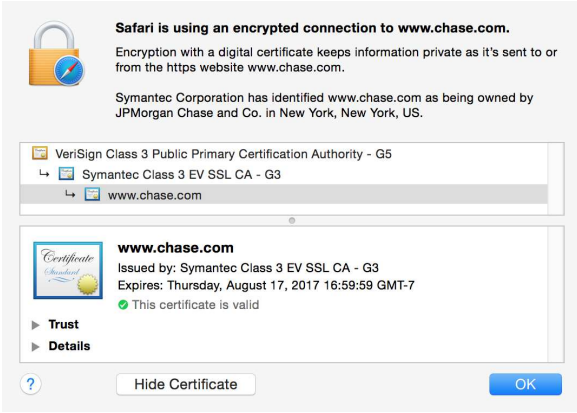
- List of certificates in the Chrome browser.

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



Digital Certificates: chase.com...

- In Safari:
  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
    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 ...
```

Digital Certificates: chase.com II

```
    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
    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
```


Digital Certificates: chase.com III

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


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.

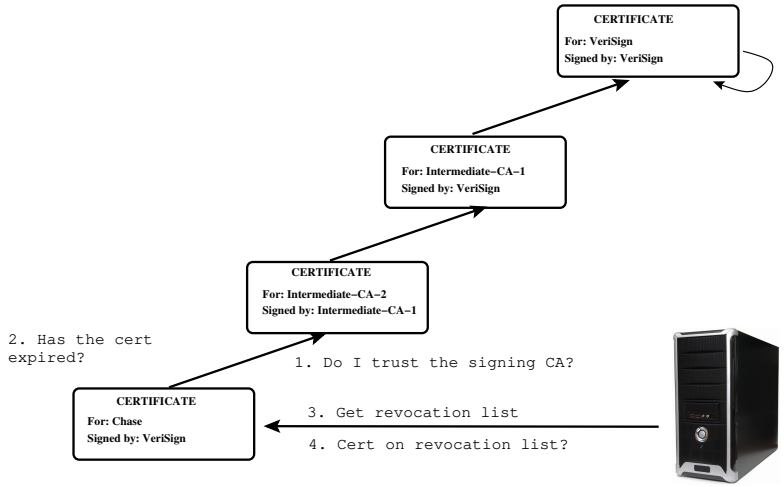
Securely connecting to chase.com

1. Alice: browses to `https://chase.com`
2. Browser: asks Chase to identify itself.
3. Chase: returns its certificate.
4. Browser:
   - Extracts $\mathrm{CA} \leftarrow$ from the certificate, signed with S_{CA}.
   - Is CA is in my pre-installed list of trusted CAs?
   - Checks if the certificate has expired?
   - Generates a session-key K.
   - Extracts $P_{\texttt{chase.com}} \leftarrow$ from the certificate.
   - Encrypts K with $P_{\texttt{chase.com}}$
   - Sends $P_{\texttt{chase.com}}(K)$ to Chase.

Certificate Hierarchy

- Certificates are signed by certificates higher in a certificate hierarchy.
- The root certificate is self-signed.
- Chain of Trust — Similar to the Trusted Platform Module's trusted boot.

Certificate Hierarchy. . .



Checking the Validity of a Certificate

- Is the certificate signed by a known trusted CA (pre-installed in the browser)?
- Has the certificate expired?
- Is the certificate revoked?
  1. Extract the revocation site URL from the certificate.
  2. Get the certificate revocation list.
  3. Is the list signed by the CA?
  4. Is this certificate serial number on the list?

Impersonation

- Eve makes a CA issue her a certificate with
  - Alice's name
  - Alice's employer
  - other real information about Alice
  - a public key Eve has generated
- Eve can now send email to Bob, appearing to be Alice.

Spoofing Updates

- In 2001 VeriSign issued 2 certificates in the name of *Microsoft Corporation*
- They could be used to trick someone into believing that a software update came from Microsoft.

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*.