CSc 466/566

Computer Security

20 : Cryptography — Signatures

Version: 2019/11/06 10:21:15

Department of Computer Science
University of Arizona

collberg@gmail.com

Christian Collberg

Outline

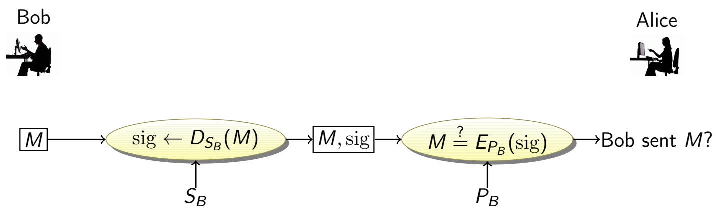
Digital Signatures

- In this lecture we are going to talk about cryptographic hash functions (checksums) and digital signatures.
- We want to be able to
  1. Detect tampering: is the message we received the same as the message that was sent?
  2. Authenticate: did the message come from who we think it came from?

Signing a Public Key

http://xkcd.com/364

Digital Signatures. . .



Why do we sign with the `decrypt` function???

- Q: Why do we sign with the `decrypt` function?
- A: We need to sign using the private key. Only the `decrypt` function takes a private key as input!

Outline

RSA Signature Scheme

1. Alice applies the decryption function to her document M with her private key S_A, thereby creating a signature $S_{\mathrm{Alice}}(M)$.
2. Alice sends M and the signature $S_{\mathrm{Alice}}(M)$ to Bob.
3. Bob applies the encryption function to the document using Alice's public key, thereby verifying her signature.

RSA Encryption: Algorithm

- Bob (Key generation):
  1. Generate two large random primes p and q.
  2. Compute $n = pq$.
  3. Select a small odd integer e relatively prime with $\phi(n)$.
  4. Compute $\phi(n) = (p-1)(q-1)$.
  5. Compute $d = e^{-1} \bmod \phi(n)$.

  - $P_B = (e, n)$ is Bob's RSA public key.
  - $S_B = (d, n)$ is Bob's RSA private key.
- Alice (encrypt a message M for Bob):
  1. Get Bob's public key $P_B = (e, n)$.
  2. Compute $C = M^e \bmod n$.
- Bob (decrypt a message C from Alice):
  1. Compute $M = C^d \bmod n$.

RSA Signature Algorithm

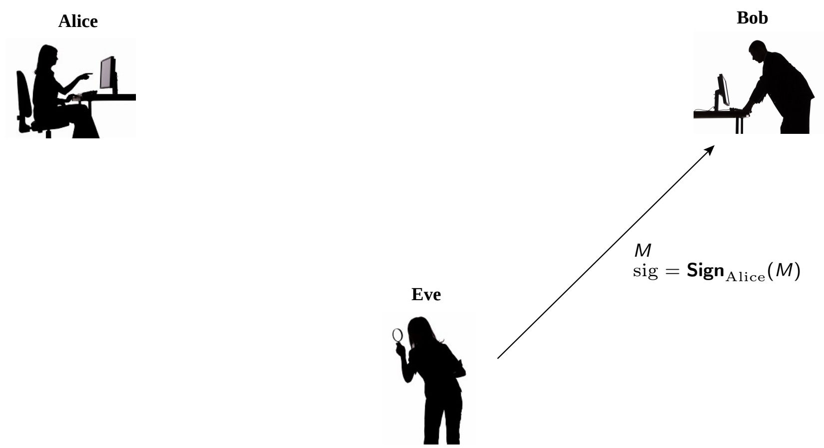
- Bob (Key generation): As before.
  - $P_B = (e, n)$ is Bob's RSA public key.
  - $S_B = (d, n)$ is Bob' RSA private key.
- Bob (sign a secret message M):
  1. Compute $S = M^d \bmod n$.
  2. Send M, S to Alice.
- Alice (verify signature S received from Bob):
  1. Receive M, S from Alice.
  2. Verify that $M \stackrel{?}{=} S^e \bmod n$.

Outline

Security Goals

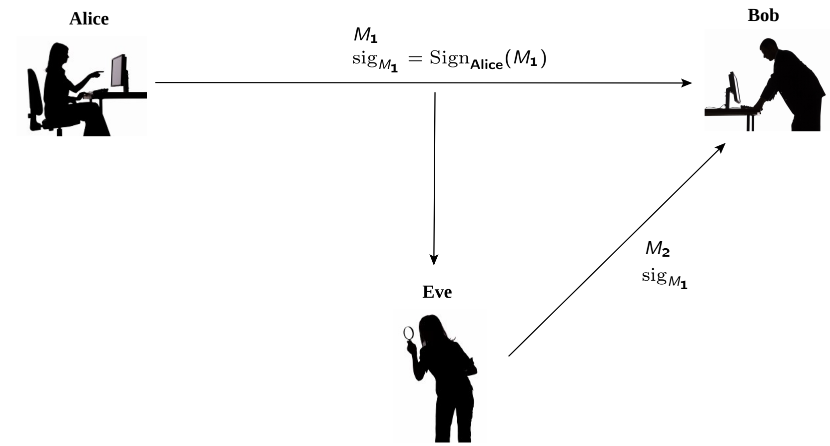
- We want to ensure:
  1. Nonforgeability
  2. Nonmutability
  3. Nonrepudiation

Nonforgeability



- Eve should not be able to create a message that appears to come from Alice.

Nonmutability



- Eve should not be able to take a valid signature for one message from Alice, and apply it to another one.

Nonrepudiation

- Alice should not be able to claim she didn't sign a document that she did sign.

Does RSA Provide Nonforgeability?

- Nonforgeability: Eve should not be able to create a message that appears to come from Alice.
- To forge a message M from Alice, Eve would have to produce

$$M^d \bmod n$$

without knowing Alice's private key d.
- This is equivalent of being able to break RSA encryption.

Does RSA Provide Nonmutability?

- Nonmutability: Eve should not be able to take a valid signature for one message from Alice, and apply it to another message.
- By itself, RSA does not achieve nonmutability.
- Not usually a problem since we normally sign the cryptographic hash of a message.

Nonmutability. . .

- Assume Eve has two valid signatures from Alice, on two messages M_1 and M_2:

$$\begin{aligned} S_1 &= M_1^d \bmod n \\ S_2 &= M_2^d \bmod n \end{aligned}$$

- Eve can then produce a new signature

$$\begin{aligned} S_1 \cdot S_2 &= (M_1^d \bmod n) \cdot (M_2^d \bmod n) \\ &= (M_1 \cdot M_2)^d \bmod n \end{aligned}$$

This is a valid signature for the message $M_1 \cdot M_2$!

Nonmutability. . .

- Does this matter?
- Yes, if Alice is signing session (symmetric) keys.
- Such keys are just random numbers.
- In that case she has just signed a new key $M_3 = M_1 \cdot M_2$!

Exercise: Goodrich & Tamassia R-8.1-4

What type of attack is Eve employing here:

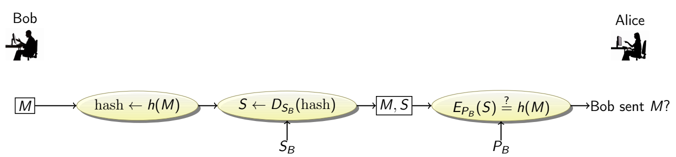
1. Eve has given a bunch of messages to Alice for her to sign using the RSA signature scheme, which Alice does without looking at the messages and without using a one-way hash function. In fact, these messages are ciphertexts that Eve constructed to help her figure out Alice's RSA private key.
2. Choose one: ciphertext only, chosen ciphertext, chosen plaintext, known plaintext.

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 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.

Signature Protocol. . .



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

Cryptographic Hash Functions. . .

- CHFs should be
  1. deterministic
  2. one-way
  3. collision-resistant

  i.e., 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.

Weak vs. Strong Collision Resistance

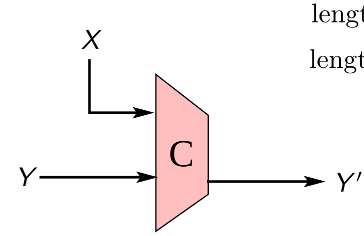
- CHFs also have the property to be collision resistant.
- Weak collision resistance:
  - Assume you have a message M with hash value $h(M)$.
  - Then it should be hard to find a different message M' such that $h(M) = h(M')$.
- Strong collision resistance:
  - It should be hard to find two different message M_1 and M_2 such that $h(M_1) = h(M_2)$.
- Strong collision resistance is hard to prove.

Merkle-Damgård Construction

- Hash functions are often built on a compression function $C(X, Y)$:

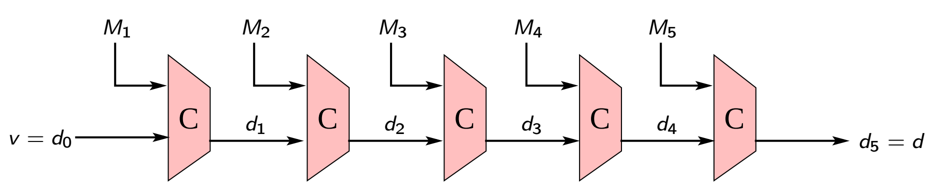
$\text{length}(X) = m$

$\text{length}(Y) = \text{length}(Y') = n$



- X is (a piece of) the message we're hashing.
- Y and Y' is the hash value we're computing.

Merkle-Damgård Construction...



- For long messages M we break it into pieces $M_1, \ldots, M_k$, each of size m.
- Our initial hash value is an initialization vector v.
- We then compress one M_i at a time, chaining it together on the previous hash value.

Outline

Textbook vs. Real World

- The textbook description of RSA is not secure.
- There are several issues that we need to think about before we use it in the real world:
  - Encrypt-then-Sign or Sign-then-Encrypt?
  - Sign and Encrypt with the Same Key?
  - Secure padding schemes.

Encrypt-then-Sign or Sign-then-Encrypt

- We often want to both sign (for authentication) and encrypt (for confidentiality) a message M. Which do we do first?
  - Sign M, then encrypt the message + its signature?
  - Encrypt M, then append the signature of M?
- Sign, then encrypt:

  http://www.cis.upenn.edu/~cse331/Fall02/Lectures/CSE331-21.pdf
- I.e: Alice wants to send a signed and encrypted message M to Bob:
  - Alice sends $E_{P_B}(M, \mathbf{sign}_{S_A}(M))$ to Bob
  - Bob first decrypts, and then checks the signature.

Sign and Encrypt with the Same Key?

- Read this

  https://www.cs.cornell.edu/courses/cs5430/2015sp/notes/rsa_sign_vs_dec.php

  to see the relationships between RSA-for-encryption and RSA-for signing.
- Summary: Use different keys for signing and encryption.

Optimal Asymmetric Encryption Padding

- We need to add padding to RSA-encrypt to make it secure to real world attacks.

  https://www.cs.cornell.edu/courses/cs5430/2015sp/notes/rsa_sign_vs_dec.php
- This is called Optimal Asymmetric Encryption Padding (OAEP):

  https://en.wikipedia.org/wiki/Optimal_asymmetric_encryption_padding

Outline

Final Exam: Protocols

Alice wants to send Bob a *very large* message M, such that

1. Bob is able to forward M to Cathy, and Cathy can verify that Alice is the originator of the message, and
2. the protocol is *efficient*, but
3. they don't care about the confidentiality of M, i.e. there's no need to encrypt M.

At their disposal, Bob and Alice have access to

1. A public key signature algorithm (RSA),
2. A cryptographic hash function (SHA-1).

Design the appropriate protocol.

Final Exam: Digital Signatures — Definitions

- Define the following terms:
  1. Nonforgeability
  2. Nonmutability
  3. Nonrepudiation

Final Exam: RSA signature: Nonmutability

- Show how the RSA signature scheme does not achieve nonmutability.
- Is this usually a problem? Why?

Final Exam: Cryptographic Hash Function Collision Resistance

- What is the difference between weak and strong collision resistance?

Final Exam: Final Exam: Merkle-Damgård Construction

- Show how, given a compression function C, a long message M can be hashed using the Merkle-Damgård Construction.

Outline

Summary

- Digital signatures make a message tamper-proof and give us authentication and nonrepudiation
- They only show that it was signed by a specific key, however
- It's cheaper to sign a checksum of the message rather than the whole message
  - Cryptographic checksums are necessary to do this securely

Readings and References

- Chapter 8.1.7, 8.2.1, 8.5.2 in *Introduction to Computer Security*, by Goodrich and Tamassia.

Acknowledgments

Additional material and exercises have also been collected from these sources:

1. Matthew Landis, *620—Fall 2003—Cryptographic Checksums and Digital Signatures*.
2. RFC1321 (MD5), `www.ietf.org/rfc/rfc1321.txt`