ELEC5471M

Data Communications and Network Security

Lectures on Network Security
Handout 2: Authentication Techniques

Mohsen Razavi

Institute of Integrated Information Systems
School of Electronic and Electrical Engineering
University of Leeds

What would network security entail?

Consider the email example. As a user, what would you expect from this service?

- We may like that nobody, but the recipient, learns about the content → confidentiality

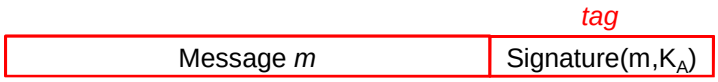
- We expect the message arrives safely with no changes in the content → message integrity
- The sender needs to make sure that the intended recipient picks up the email; similarly, the recipient needs to make sure that the message is genuinely sent by the declared sender → authentication

In the next two lectures, we will see how we can achieve message integrity and end-point authentication

- In daily life, how do you authenticate a message?

Digital Signature

- In normal life, your signature serves two purposes: You certify that you agree with the content of the signed document (message integrity) and that this is YOU, who agrees with the content (authentication).
- It would be great if we can devise a *digital signature* technique that can also tick both objectives in one go!
- Such a signature must be message dependent (to certify its integrity), as well as sender/receiver dependent (to authenticate that party). Let's authenticate Alice.

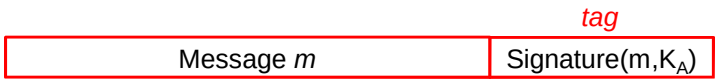


  - If we need confidentiality as well, we can encrypt the whole record. For now we only focus on the authentication problem. Whether we need confidentiality or not, it is application dependent.

- What other properties our digital signature should have?

Digital Signature: Properties

- Digital signature must be message dependent (to certify its integrity), as well as sender/receiver dependent (to authenticate that party). Let's authenticate Alice.



- Digital signature must be verifiable! That is knowing *m*, Bob should be able to verify the *tag* is genuinely generated by Alice.
- But digital signature must be non-forgeable! That is nobody can change the message *m* and find the corresponding *tag* to the altered message.
- It should ideally be binding! That is Bob could take the Alice message to court, and prove that is hers.
- Any ideas?

Digital Signature

- A digital signature must be message dependent, sender/receiver dependent, verifiable but non-forgeable, and binding

Message *m*	*tag* Signature(m,K_A)

- So far, we have learned about *symmetric key* and *public key* systems.
  - What can we do if Alice and Bob share a secret key?
  - What can we do if Alice and Bob each have a public key known to the other party?
- If we want that even the legitimate receiver cannot forge our signature (not always the case, but suppose you are dealing with a non-trustworthy seller), then can we sign with a symmetric key shared between the two users?
- How about using public keys?

RSA: Another important property

Remember:

$$\underbrace{K_B^-(K_B^+(m))}_{\text{use public key first, followed by private key}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{use private key first, followed by public key}}$$

result is the same!

Follows directly from modular arithmetic:

$(m^e \bmod N)^d \bmod N = m^{ed} \bmod N$

$= m^{de} \bmod N$

$= (m^d \bmod N)^e \bmod N$

Does it give you any idea?

Digital Signature using Public Keys

- Suppose Alice and Bob each have a public key known to the other party. Alice can sign using her private key:

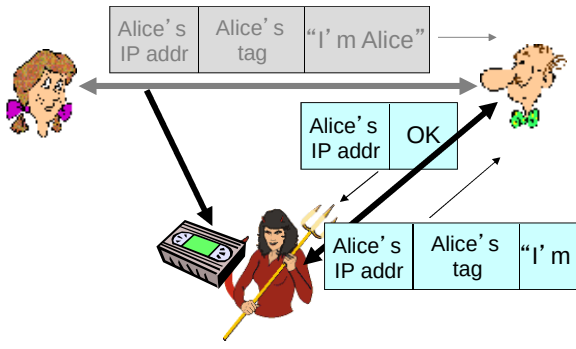
Message *m*	*tag* $K^{-}_{A}(m)$

- Is it message dependent?
- Alice dependent?
- Verifiable?
- non-forgeable?
- Binding?

- Yes.
- Yes, only Alice knows K^{-}_{A}
- Yes, Bob knows the public key of Alice: K^{+}_{A}. He just needs to verify $K^{+}_{A}(tag) = m$.
- Yes, if someone changes m to m', she has to change the tag to K^{-}_{A} but nobody but Alice knows K^{-}_{A}.
- Yes, because no one but Alice could have signed it. *But is that so?*

Playback Attack

- Trudy can record Alice conversation and plays it back at some time later. For instance, you place an order today, but then somebody keeps ordering the same thing for you!



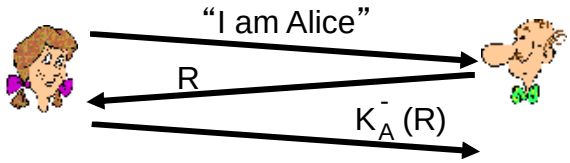
playback attack: Trudy records Alice's packet and later plays it back to Bob

- How can it be avoided?

Playback Attack: Counterattack

- The main counterattack idea is to leave some time dependent information in the message.
- We have seen similar scenarios in the networking section. What did we use then?
- Another solution is to use a nonce

nonce: number (R) used only *once-in-a-lifetime*



Bob computes

$K_A^+(K_A^-(R)) = R$

and knows only Alice could have the private key to encrypt R s.t.

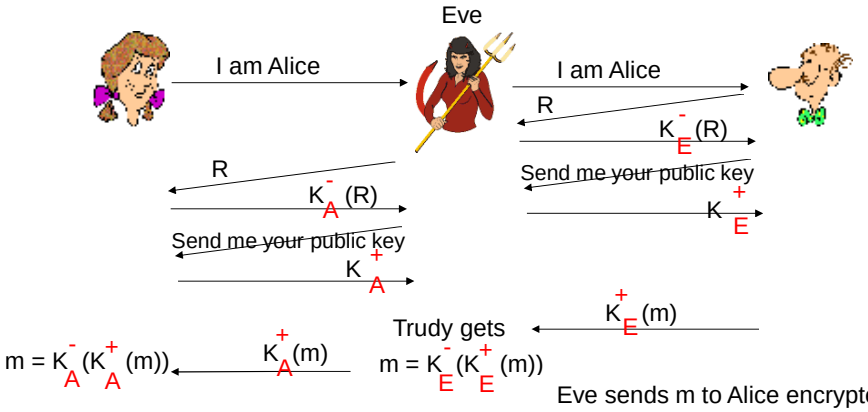
$K_A^+(K_A^-(R)) = R$

- Nonce is typically used to authenticate a live session, whereas sequence numbers can be used in authenticating live packets

(Wo)Man-in-the-middle Attack

- We had a key assumption in our developing the digital signature: Alice and Bob each have a public key known to the other party. But, how can we guarantee that? What if we can't?

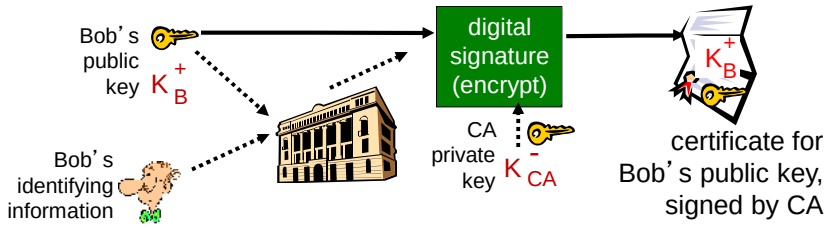
man (or woman) in the middle attack: Eve poses as Alice to Bob, and as Bob to Alice. Then she can eavesdrop without being detected



Eve sends m to Alice encrypted with Alice's public key

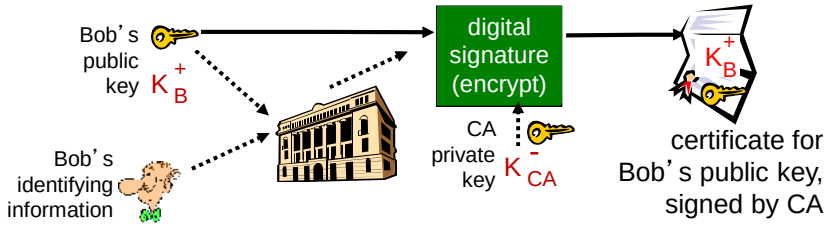
Certification Authorities

- We had a key assumption in our developing the digital signature: Alice and Bob each have a public key known to the other party. But, how can we guarantee that?
- *Certification authority (CA):* binds public key to a particular entity, such as Bob.
- Bob (person, router) registers its public key with CA.
  - Bob provides "proof of identity" to CA.
  - CA creates a certificate, binding Bob to its public key.
  - certificate contains Bob's public key and some information about the identity of bob, both digitally signed by the CA.



Certification Authorities

- *Certification authority (CA):* binds public key to a particular entity, such as Bob.
- Now, whenever Alice needs Bob's public key, she asks Bob for its certificate. She will then apply K^+_{CA} to the certificate to find the Bob's public key.
- We now need to know K^+_{CA}. This is typically provided in your trusted web browser. After all, it seems that we have to trust someone (but not everyone)!



- Have we now solved all problems around authentication? Well, almost!

Reminder: Encryption in practice

- Flashback: Challenges of ~~one-time pad:~~
  - *digital signature*
  - 1- How Alice and Bob share a secret key to begin with? Do they have to come together every time? Remember that some of Alices and Bobs are simply machines, routers, …
  - 2- One-time pad needs a key as long as the message. Is the price of losing some security, can we devise with protocols that are practically secure and more efficient in the usage of the precious key?

Message *m*	*tag* $K_A^-(m)$

- Public-key can solve the first problem, but still the exponentiation needed in RSA is computationally intensive.
- It would be more practical if we find efficient ways for ~~encrypting data~~ even if they need a symmetric key.
  - *authentication*
- Idea: We can generate the required session keys using RSA, and then use relevant symmetric schemes for ~~encryption/decryption~~.
  - *authentication/message integrity*

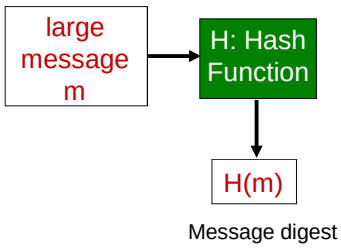
FACULTY OF ENGINEERING UNIVERSITY OF LEEDS

Hash Functions

Problem: Computationally expensive to public-key-encrypt long messages

Goal: fixed-length, easy- to-compute digital "fingerprint"

Solution: Apply hash function H to *m*, get fixed size message digest, *H(m).*



Message digest

Hash function properties:

- many-to-1
- produces fixed-size msg digest (fingerprint)
- given message digest x = H(m), computationally infeasible to find m' such that x = H(m')

Hash function algorithms:

- MD5 (RFC 1321)
  - computes 128-bit message digest in 4-step process.
- SHA-1
  - US standard [NIST, FIPS PUB 180-1]
  - 160-bit message digest

FACULTY OF ENGINEERING 14 UNIVERSITY OF LEEDS

Message Authentication Code (MAC)

- So, how about this signature?

Message *m*	*tag* H(m)

It is only a function of m; everyone who knows H, can replace m with something else!

- How about this one?

Message *m*	*tag* $K^-_A(H(m))$

It looks good; It is provably Alice's signature and we have only applied the public key to a small number. No one can forge the signature (including Bob).

- What if Alice and Bob share a secret key S?

Message *m*	*MAC* H(m+S)

If we trust Bob (or if we handle a naughty Bob in some other layer), then it is good too. So long as Eve does not know S, she cannot forge the message. Why?

- The required secret key S can be exchanged at the beginning of the session using public key cryptography → SSL

Summary of ideas used in authentication

- So, authentication is possible via the following techniques:
  - Public-key signature: the private key of each user (K^-) can be used to digitally sign the message
  - Certification authority (CA): We can verify the public key of any user registered at the CA. Each user has its own certificates.
  - Nonce and sequence number: To certify that the session is live, and it is not a playback attack
  - Message digest: to apply a Hash function to the message to get a fixed-size output.
  - Message authentication code (MAC): A message digest for a combination of a message and a secret key. It provides message integrity and authentication for two (trusted) users.
  - Sharing a secret key using public-key cryptography

Let's apply all these in practice. Next, we look at how security is achieved in emails and at the transport layer (SSL).