COMx501: Computer Security and Forensics

Achim D. Brucker

a.brucker@sheffield.ac.uk **https://www.brucker.ch/**

Software Assurance & Security Research
Department of Computer Science, The University of Sheffield, Sheffield, UK
https://logicalhacking.com/

February 21, 2018

```
Intent i = ((CordovaActivity) this.cordova.getActivity()).getIntent();
String extraName = args.getString(0);
if (i.hasExtra(extraName)) {
    callbackContext.sendPluginResult(new PluginResult(PluginResult.Status.OK, i.getStringExtra(extraName)));
    return true;
} else {
    callbackContext.sendPluginResult(new PluginResult(PluginResult.Status.ERROR));
    return false;
}
```

COMx501: Computer Security and Forensics

Part 6: Signatures and PKIs

Achim D. Brucker

a.brucker@sheffield.ac.uk **https://www.brucker.ch/**

Software Assurance & Security Research
Department of Computer Science, The University of Sheffield, Sheffield, UK
https://logicalhacking.com/

February 20, 2017 (updated February 21, 2018)

Outline

1 Introduction

2 Digital Signatures

3 Public Key Infrastructures (PKI)

4 Further Topics: Revocation and Recovery

5 Conclusion

6 Appendix

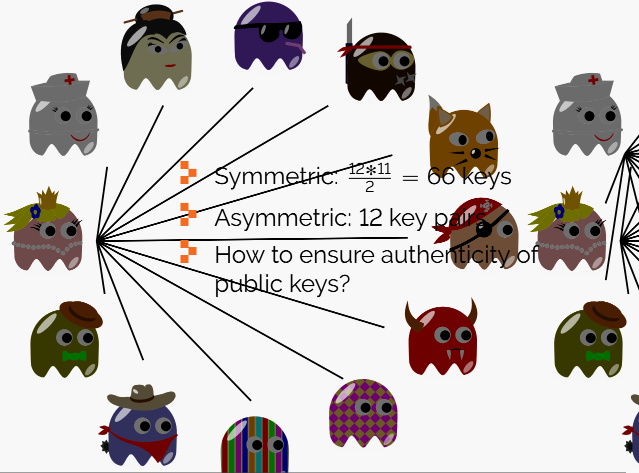
Motivation

> “ Public-key cryptography was born in May 1975, the child of two problems: **the key distribution problem** and **the problem of signatures**. The discovery consisted not of a solution, but of the recognition that the two problems, each of which seemed unsolvable by definition, could be solved at all and that the solutions to both came in one package.
>
> Whitfield Diffie, *The first-ten years of public-key cryptography*, 1988

- Today: to what extend are these problems solved?
- Recommendation:
  Information Security: Before & After Public-Key Cryptography by Whitfield Diffie
  https://youtu.be/1BJuuUxCaaY

The Key Distribution Problem

How to provide a strong link between a key and an identity?



- Symmetric: $\frac{12*11}{2} = 66$ keys
- Asymmetric: 12 key pairs
- How to ensure authenticity of public keys?

Outline

1 Introduction

2 Digital Signatures

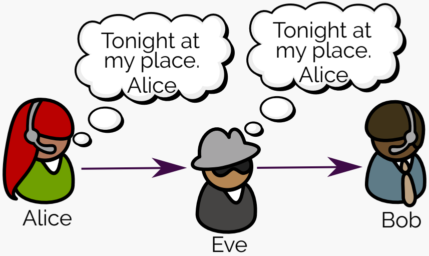
3 Public Key Infrastructures (PKI)

4 Further Topics: Revocation and Recovery

5 Conclusion

6 Appendix

The Digital Signature Problem



- Problem of proof of data origin.
  How do we know, or prove, that a message originated from a particular person?
- Public-key cryptography supports both knowing and proving to others (non-repudiation).
  Would this be possible using a shared key?

Digital Signature Requirements

- Signature is fundamental in authentication and non-repudiation.
- Nomenclature and set-up:
  - $\mathcal{M}$ is set of messages that can be signed.
  - $\mathcal{S}$ is set of elements called signatures, e.g. n-bit strings.
  - $S_A : \mathcal{M} \to \mathcal{S}$ is a signing transformation for entity A, and kept secret by A.
  - $V_A : \mathcal{M} \times \mathcal{S} \to \{\text{true}, \text{false}\}$ is a verification transformation for A's signature and is publicly known.
- S_A and V_A provide digital signature scheme for A.

Signature Schema

- Example:

$$\begin{array}{llll}
S_A(m_0) = s_0 & V_A(m_0, s_0) = \text{true} & V_A(m_1, s_0) = \text{false} & V_A(m_2, s_0) = \text{false} \\
S_A(m_1) = s_1 & V_A(m_0, s_1) = \text{false} & V_A(m_1, s_1) = \text{true} & V_A(m_2, s_1) = \text{false} \\
S_A(m_2) = s_2 & V_A(m_0, s_2) = \text{false} & V_A(m_1, s_2) = \text{false} & V_A(m_2, s_2) = \text{true}
\end{array}$$

- **Signing procedure**: A creates a signature for $m \in \mathcal{M}$ by:
  Compute $s = S_A(m)$ and transmit pair (m, s).
- **Verification procedure**. B verifies A's signature of (m, s) by:
  Compute $u = V_A(m, s)$. Accept signature only if $u = \textit{true}$.
- **Important:**
  e it is hard for any entity other than A to find, for any $m \in \mathcal{M}$, an $s \in \mathcal{S}$, where $V_A(m, s) = \textit{true}$.

Implementing Digital Signatures (1/2)

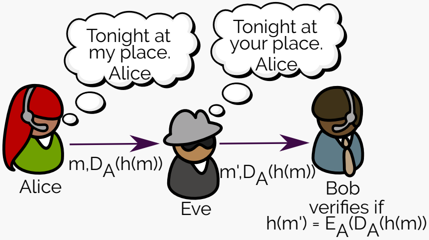
- Can be based on (reversible) public-key encryption systems.
- Consider $E_e : \mathcal{M} \rightarrow \mathcal{C}$ is a public-key transformation. Moreover, suppose that $\mathcal{M} = \mathcal{C}$.
  If D_d is the decryption transformation corresponding to E_e, then since both are permutations

  $$D_d(E_e(m)) = E_e(D_d(m)) = m \qquad \text{for all } m \in \mathcal{M}.$$

  A public-key encryption scheme of this type is called reversible.
- Construction for a digital signature schema
  1. Let $\mathcal{M}$ and $\mathcal{C}$ be message and signature space, with $\mathcal{M} = \mathcal{C}$.
  2. Let (e,d) be a key pair for the public-key encryption scheme.
  3. Define signing function S_A to be D_d. I.e., $s = D_d(m)$.
  4. Define V_A by

  $$V_A(m,s) = \begin{cases} \text{true}, & \text{if } E_e(s) = m, \\ \text{false}, & \text{otherwise} \end{cases}$$

Implementing Digital Signatures (2/2)


Tonight at my place. Alice
Tonight at your place. Alice
Alice
m,D_A(h(m))
Eve
m',D_A(h(m))
Bob verifies if h(m') = E_A(D_A(h(m))

- RSA provides a realization of digital signatures

$$D_d(E_e(m)) = E_e(D_d(m)) = m$$

- To prevent forgery, sign messages with fixed structure, e.g.:
  - message names its sender, or (more typically)
  - cryptographic hash signed, sent with the message

Pair can additionally be encrypted for confidentiality.

Outline

1 Introduction

2 Digital Signatures

3 Public Key Infrastructures (PKI)

- Introduction
- An Example: X.509
- Trust Models

4 Further Topics: Revocation and Recovery

5 Conclusion

6 Appendix

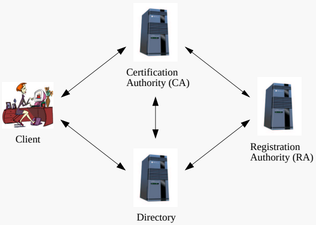
The Essence of Public Key Infrastructure (PKI)

A PKI is an infrastructure that allows principals to recognize which public key belongs to whom (i.e. to bind public keys to principals).

- To join the PKI, Alice
  - generates her own public/private key pair,
  - takes her public key K_A to a certification authority (CA) that everybody trusts and says "I am Alice and K_A is my public key".
- The CA verifies that Alice is who she says she is, and then signs a digital certificate that states

  "K_A is Alice's public key".
- Now
  - Any principal, e.g. Bob, can now check the certificate to obtain Alice's public key K_A and accept it as valid.
  - Alice can similarly obtain Bob's public key K_B.
- Thus, the CA can help to establish mutual trust

PKI services and components



- PKI services:
  - Linking public keys to entities (certificates).
  - Key life-cycle management (key revocation, recovery, updates).
- PKI components:
  - Certification Authority (CA):
    - Creates certificates and publishes them in directory.
    - Maintains Certificate Revocation List (CRL) in directory.
      CRL checked actively by single clients or by validation services.
    - Backs up certain keys (for key recovery or escrow).
  - Directory:
    - Makes user certificates and CRLs available.
    - Must identify users uniquely (needs fresh/accurate user data).
    - Must be highly available.
  - Registration Authority (RA):
    - Manages process of registering users and issuing certificates.
    - Ensures proper user identification.
  - Clients: different uses of a PKI, e.g.
    - authentication (one-way, two-way, or three-way),
    - signed documents and transactions.

Certificates

- A certificate is a token that binds an identity to a key.

Example

Issuer CA “Cathy” signs a hash of the identity of the principal to whom the certificate is issued (Alice), the corresponding public key (K_{Alice}), and information such as time of issue or expiration (T):

$$\{h(K_{\mathrm{Alice}},\ \text{Alice},\ T)\}_{K^{-1}_{\text{Cathy}}}$$

- To validate the certificate, a principal Bob must obtain the issuer’s public key and use it to decipher the hash and check the data in the certificate.
- To illustrate certificates and certification, let us consider a concrete example: X.509.

Outline

1 Introduction

2 Digital Signatures

3 Public Key Infrastructures (PKI)
- Introduction
- An Example: X.509
- Trust Models

4 Further Topics: Revocation and Recovery

5 Conclusion

6 Appendix

X.509

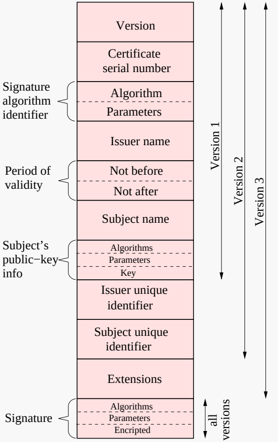
- A standard, part of the X.500 series of ITU-T recommendations, that defines a framework for authentication services.
- The certificate structure (certificate formats and certification validation) and authentication protocols defined in X.509 are used in a variety of contexts, e.g. in IPSEC, SSL/TLS, and S/MIME.
- It is based on public-key cryptography (it recommends RSA), hashes, and digital signatures.
- The heart of the X.509 scheme is the public-key certificate associated with each user, which is created by the CA and is placed in the directory by the CA or by the user.

X.509 Certificate Components

Serial number must be unique among the certificates issued by this issuer.
I.e. pair *(issuer name, serial number)* must be unique.

Signature algorithm identifier of algorithm, and any parameters, used to sign the certificate.

Issuer name is X.500 name of CA that created and signed this certificate.
Optional string issuer unique identifier in the event the X.500 name has been reused for different entities.



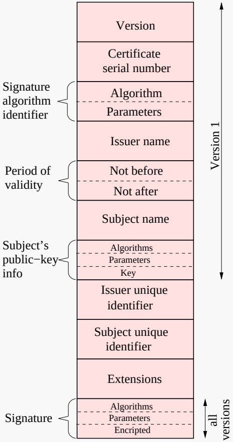
X.509 Certificate Components (cont.)

Period of validity.

Subject name is the name of the user to whom the certificate refers (i.e. the user whose public key is certified). Optional bit string subject unique identifier in the event the X.500 name has been reused for different entities.

Subject public-key info identifies the algorithm, its parameters, and the subject's public key.

Signature contains the hash code of the other fields, encrypted with the CA's private key.

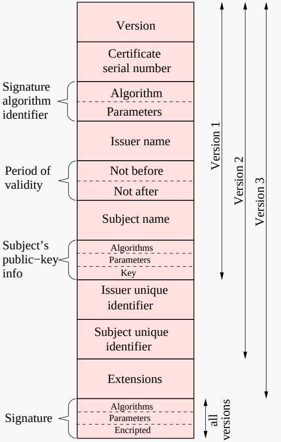


The X.509 certificate

The certificate of user A issued (and signed with K_{CA}^{-1}) by CA is

$$\begin{aligned} CA \ll A \gg = & \ (V, SN, AI, CA, T_A, A, Ap), \\ & \{h(V, SN, AI, CA, T_A, A, Ap)\}_{K_{CA}^{-1}} \end{aligned}$$

- To validate $CA \ll A \gg$, and verify the user public key that was generated, Bob obtains CA's public key for the particular signature algorithm and deciphers the signature.
- Bob then uses the information in the signature field to recompute the hash value from the other fields. If it matches the deciphered signature, the signature is valid if the issuer's public key is correct.
- Bob then checks the period of validity to ensure that the certificate is current.



Outline

1 Introduction

2 Digital Signatures

3 Public Key Infrastructures (PKI)

- Introduction
- An Example: X.509
- Trust Models

4 Further Topics: Revocation and Recovery

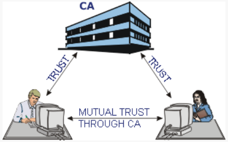
5 Conclusion

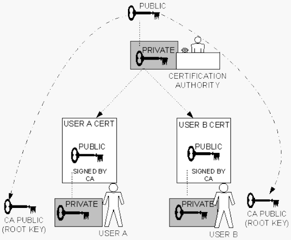
6 Appendix

Trust models: Direct Trust

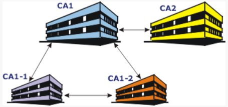
Direct trust:

- If all users subscribe to the same CA, then there is a common trust of that CA.
- All user certificates can be placed in the same directory for access by all users.





Trust Models: Hierarchical Trust

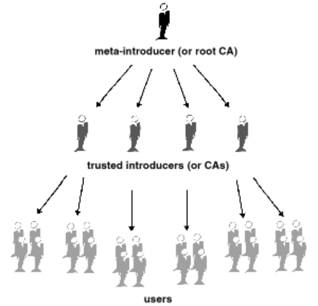


Direct trust:

- For a large community of users, it is more practical to have a number of CA's, each of which securely provides its public key to some fraction of the users.

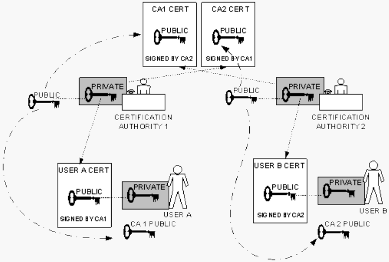
Trust tree:

- Trust extends from a number of root certificates.
- These certificates may certify certificates themselves, or they may certify certificates that certify still other certificates down some chain.
- The leaf certificate's validity is verified by tracing backward from its certifier, to other certifiers, until a directly trusted root certificate is found.



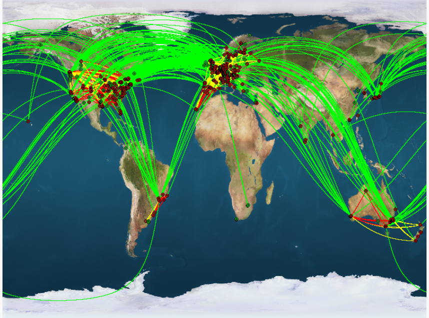
Hierarchical Trust and Cross-Certification

- Suppose that A and B have obtained certificates from CAs X_1 and X_2, respectively. If A does not securely know X_2's public key, then A cannot validated B's certificate.
- Cross-certification: if the CAs have exchanged their own public keys, then A can obtain B's public key by a chain of certificates.
  - A obtains, from the directory, the certificate of X_2 signed by X_1.
    A can thus get hold of X_2's public key (and verify it by means of X_1's signature on the certificate).
  - A then goes back to the directory and obtains the certificate of B signed by X_2, which A can now verify with the trusted copy of X_2's public key.
- X.509 suggests arranging CAs in a hierarchy.



Trust models: Web of Trust

- Web of trust:
  - encompasses direct and hierarchical trust,
  - adds the ideas that trust is in the eye of the beholder (which is the real-world view) and that more information is better.
- A certificate is trusted directly, or trusted in some chain going back to a directly trusted root certificate (the meta-introducer), or by some group of introducers.

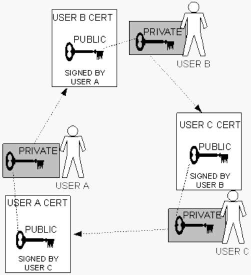


Web of Trust: PGP/GnuPG

- Pretty Good Privacy (PGP): an encipherment program widely used to provide privacy for e-mail and to sign files digitally.
- It uses a certificate-based key management infrastructure for users' public keys.
- PGP certificates (and key management) differ from X.509 certificates in several important ways, e.g.
  - A PGP key may have multiple signatures (even "self-signing").
    Each user creates and signs certificates for the people he or she knows (hence, no need for central infrastructure).
  - A notion of "trust" is embedded in each signature, and the signatures for a single key may have different levels of trust
    (and the users of a certificate act according to trust level).

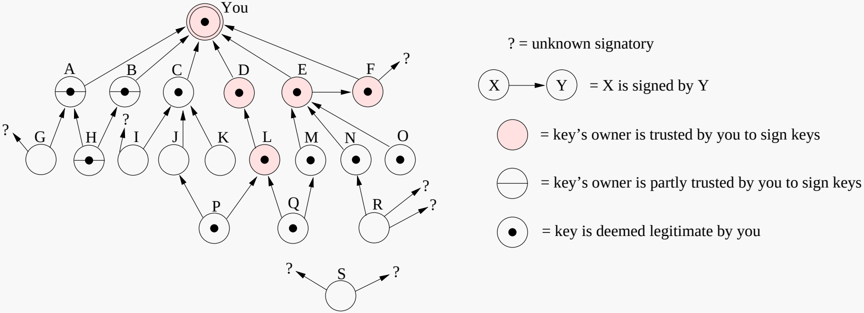
Web of Trust: PGP/GnuPG (cont.)

- In a PGP environment, any user can act as a certifying authority.
  - Digital signatures as form of introduction: when any user signs another's key, he or she becomes an introducer of that key.
  - As this process goes on, it establishes a web of trust.
- Any PGP user can validate another user's public key certificate, but such a certificate is only valid to another user if he recognizes the validator as a trusted introducer.
  - I.e. you trust my opinion that others' keys are valid only if you consider me to be a trusted introducer.
  - Otherwise, my opinion on other keys' validity is unimportant.



Web of Trust: PGP/GnuPG (cont.)

- Stored on each user's public keyring are indicators of
  - whether or not the user considers a particular key to be valid,
  - the level of trust the user places on the key that the key's owner can serve as certifier of others' keys.
- You indicate, on your copy of my key, whether you think my judgement counts. It's really a reputation system: certain people are reputed to give good signatures, and people trust them to attest to other keys' validity.



? = unknown signatory

X → Y = X is signed by Y

○ = key's owner is trusted by you to sign keys

⊖ = key's owner is partly trusted by you to sign keys

⊙ = key is deemed legitimate by you

Outline

1 Introduction

2 Digital Signatures

3 Public Key Infrastructures (PKI)

4 Further Topics: Revocation and Recovery

5 Conclusion

6 Appendix

PKI -- Key/Certificate Revocation

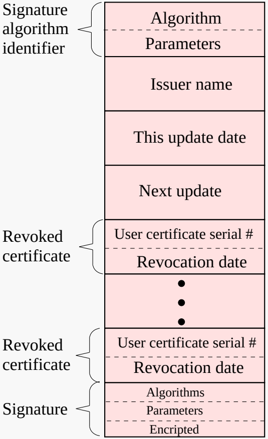
- Certificate Revocation List (CRL) signed and maintained by CA.
  - Posted on the directory.
  - Either clients check themselves actively (also with local caches), or use validation service that collects and checks CRLs centrally.
  - Each CA maintains a list of all revoked but not expired certificates issued by that CA (both to users and to other CAs).
- Reasons for revocation:
  - The user's private key is assumed to be compromised.
  - The user is no longer certified by the CA.
  - The CA's certificate is assumed to be compromised.
- X.509:
  - Each certificate includes a period of validity.
  - Typically, a new certificate is issued before the old one expires.

X.509 Certificate Revocation

Each CRL includes:

- the issuer's name,
- the date the CRL was created,
- the date the next CRL is scheduled to be issued,
- an entry for each revoked certificate.

CRL needs to be consulted for each certificate validation



Signature algorithm identifier

Algorithm

Parameters

Issuer name

This update date

Next update

Revoked certificate

User certificate serial #

Revocation date

•
•
•

Revoked certificate

User certificate serial #

Revocation date

Signature

Algorithms

Parameters

Encripted

PKI -- Key Recovery

- How can one recover a key that is lost, or if the people who know it are unable or unwilling to reveal it?
  Important, e.g., for keys belonging to roles.
- Three alternatives: either the key or the cryptosystem is weak, or a copy of the key can be placed somewhere.
- A key escrow system is a system in which a third party can recover a cryptographic key.
  - For business (e.g. recovery of backup keys),
  - or law enforcement (recovery of keys used to encipher communications to which an authority requires access, such as enciphered letters or telephone messages).

Outline

1 Introduction

2 Digital Signatures

3 Public Key Infrastructures (PKI)

4 Further Topics: Revocation and Recovery

5 Conclusion

6 Appendix

Conclusion

Digital signature schemes

- Provide
  - authentication
  - non-repudiation
- Help to solve the “key distribution” problem
- Can be implemented using reversible public-key crypto systems (e.g., RSA)

Thank you for your attention!
Any questions or remarks?

Contact:

Dr. Achim D. Brucker
Department of Computer Science
University of Sheffield
Regent Court
211 Portobello St.
Sheffield S1 4DP, UK

a.brucker@sheffield.ac.uk
@adbrucker
https://de.linkedin.com/in/adbrucker/
https://www.brucker.ch/
https://logicalhacking.com/blog/

Bibliography I

Ross J. Anderson.
Security Engineering: A Guide to Building Dependable Distributed Systems.
John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 2001.
The complete book is available at: http://www.cl.cam.ac.uk/~rja14/book.html.

Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot.
Handbook of Applied Cryptography.
CRC Press, Inc., Boca Raton, FL, USA, 5th edition, 2001.
The complete book is available at: http://cacr.uwaterloo.ca/hac/.

Bruce Schneier.
Applied Cryptography.
John Wiley & Sons, Inc., 2 edition, 1996.

Document Classification and License Information

- This presentation is classified as *Student (COMx501 – 2017/18)*:
  Except where otherwise noted, this presentation is classified "*Student (COMx501 – 2017/18)*" and only available to students of the University of Sheffield that are registered to the module "COMx501: Computer Security and Forensics" in the academic year 2017/2018. Disclosure to third parties only after a confidentiality agreement has been signed.