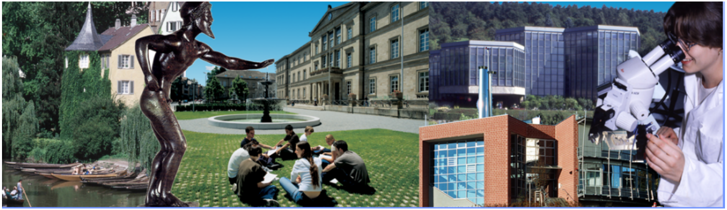
Eberhard Karls Universität Tübingen



Introduction to Computer Security

Hash Functions and Digital Signatures

Pavel Laskov

Wilhelm Schickard Institute for Computer Science

Integrity objective in a wide sense

- Reliability
  - Transmission errors
  - Corruption of stored data
- Security
  - Manipulation of data in transmission
  - Manipulation of stored data

Integrity checking: a general framework

- Compute a “digest” for the original message $D = h(M)$, such that

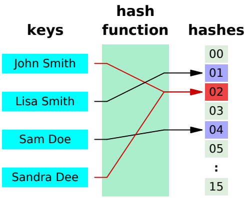
$$P(D \text{ is corrupt}) \ll P(M \text{ is corrupt})$$

.

- To check the integrity of a message M' at a later time, compute $D' = h(M')$ and verify that $D = D'$.

Hash functions

- Hash function converts large, variable size input into small, fixed size output.
- Applications:
  - Efficient search (hash tables)
  - Indexing of variable size data
  - Finding duplicate entries
  - Finding similar entries



- Requirements:
  - Efficiency: less that $\log_2 n$ comparisons
  - Determinism: always maps the same input to the same output
  - Small probability of collisions
  - Uniformity: equal probability of output values

Hash function design

- Fixed length (numeric) keys:
  - Division method:

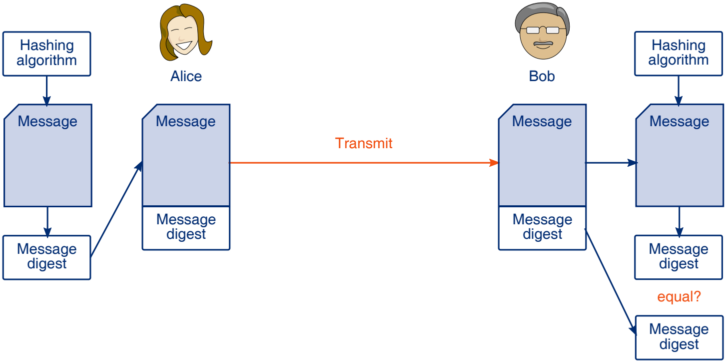
$$h(k) = k \mod m$$

  - Multiplication method:

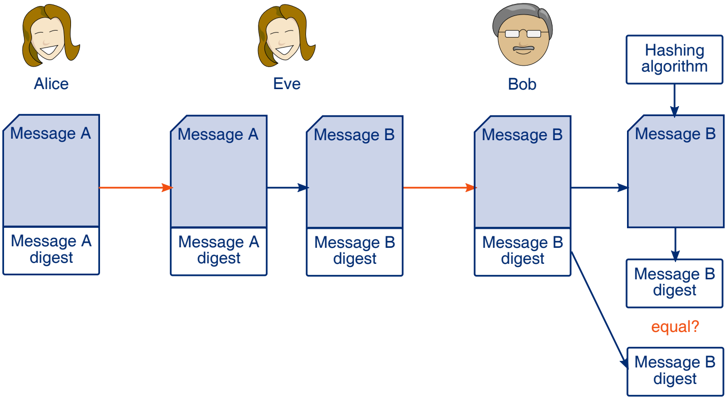
$$h(k) = \lfloor m(kA \mod 1) \rfloor$$

- Variable length keys (e.g. strings):
  - Convert a string into a fixed number (e.g. add up all characters)
  - Compute a hash of a fixed number
- Hash function reduces the dimension of the key set:
  - Collisions are unavoidable!

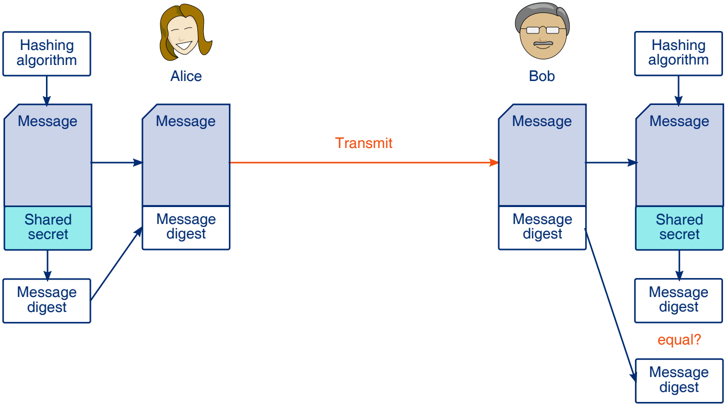
A simple message digest application



Insecurity of message digest



Message authentication code (MAC)



Secure hash function requirements

- Compression: h reduces M to a fixed size.
- For any M, $h(M)$ is easy to compute.
- Preimage resistance: For any value D, it is computationally infeasible to find M such that $D = h(M)$.
- Second preimage resistance: For any values D and M such that $D = h(M)$, it is computationally infeasible to find $M' \neq M$ such that $D = h(M')$.
- Collision resistance: It is computationally infeasible to find any pair M_1, M_2 such that $h(M_1) = h(M_2)$.

A birthday paradox

- How many people must be in a room for someone to have the same birthday as you?

A birthday paradox

- How many people must be in a room for someone to have the same birthday as you? (365)

A birthday paradox

- How many people must be in a room for someone to have the same birthday as you? (365)
- How many people must be in a room for some people to have the same birthday?

A birthday paradox

- How many people must be in a room for someone to have the same birthday as you? (365)
- How many people must be in a room for some people to have the same birthday? (23)

A birthday paradox

- How many people must be in a room for someone to have the same birthday as you? (365)
- How many people must be in a room for some people to have the same birthday? (23)
- Probability that n people have different birthdays is:

$$\bar{P}(n) = 1 \times \left(1 - \frac{1}{365}\right) \times \left(1 - \frac{2}{365}\right) \times \ldots \times \left(1 - \frac{n-1}{365}\right)$$
$$\approx \left(e^{-\frac{1}{365}}\right) \times \left(e^{-\frac{2}{365}}\right) \times \ldots \times \left(e^{-\frac{n-1}{365}}\right)$$
$$= e^{-\left(\frac{1}{365} + \frac{2}{365} + \ldots \frac{n-1}{365}\right)} \approx e^{-\frac{n^2}{2 \cdot 365}}$$

- Solving for n and equating to 0.5, we obtain:

$$n = \sqrt{2 \ln 2} \cdot \sqrt{365} = 22.54$$

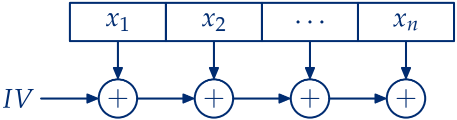
Brute force attacks

For an ideal hash function with the output size n, it should take

- 2^n operations to stage a second-preimage attack,
- $2^{n/2}$ operation to stage a collision attack.

A cryptographic strength of a hash function strongly depends on its output size.

A simple hash function



- Fix an initialization value IV.
- Compute intermediate states

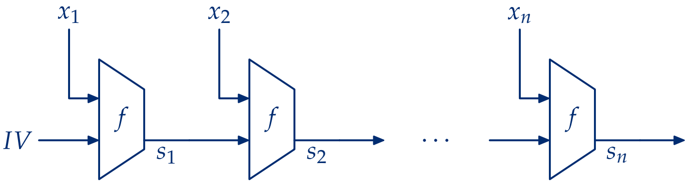
$$
\begin{aligned}
s_1 &= IV + x_1 \\
s_i &= s_{i-1} + x_i, \quad \forall i = 2, \ldots, n
\end{aligned}
$$

- Output the final state s_n as a hash value

Insecurity of a simple hash function

- Suppose the digest D is known.
- The IV can be found by sending the word $(0, 0, \ldots, 0)$.
- Then the message $(IV, D, 0, \ldots, 0)$ will produce the hash value D: the second preimage property is broken!

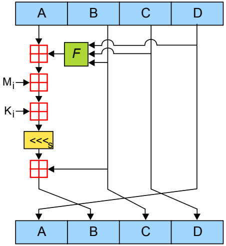
General design of a hash function



- Iterated application of a compression function $s_i \leftarrow f(x_i, s_{i-1})$
- s_0 is initialized to some fixed IV
- Collision resistance of f implies collision resistance of a hash function (Merkle's principle).

MD5 algorithm

- Pad a message to a the length $448 \mod 512$, add message length as a 64-bit value.
- Initialize 4 32-bit registers A, B, C, D with pre-defined values.
- Divide each 512-bit block in 16 words w of length 32; for $i = 1 \ldots 64$, do:
  - compute $A + f_i(B, C, D)$
  - add $M_i = w_{g_i}$
  - add $K_i = \lfloor |\sin(i+1)| \cdot 2^{32} \rfloor$
  - shift left by s_i positions
  - Add B and save in B, $C \leftarrow B$, $D \leftarrow C$, $A \leftarrow D$
- Proceed to the next block using the state A, B, C, D.



Insecurity of MD5

[1996]: H. Dobbertin demonstrated a collision of the compression function of MD5.

[2004]: X. Wang and H. Yu showed a modular differential attack on the complete MD5 hash function.

[2007]: M. Stevens, A. Lenstra and B. de Weger demonstrated a chosen prefix attack: given a prefix P, find suffixes S_1 and S_2 such that MD5 hashes of $P||S_1$ and $P||S_2$ are the same.

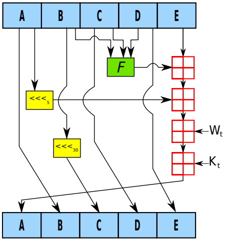
[2008]: A. Sotirov et al. generated a rogue X.509 certificate to demonstrate practical consequences of MD5 vulnerability.

SHA-1 algorithm

- 160-bit output instead of 128 in MD5; 80 rounds per 512-bit block.
- Similar initialization and overall scheme.
- 16 initial words w_i are extended into the total of 80 words, one per round:

$$w_i = (w_{i-3} \oplus w_{i-8} \oplus w_{i-14} \oplus w_{i-16}) \lll 1$$

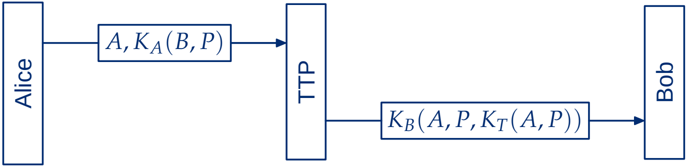
- Fixed constants and shifts per round.
- Best known collision attack requires 2^{69} operations, compared to 2^{80} by brute force.



Authentication

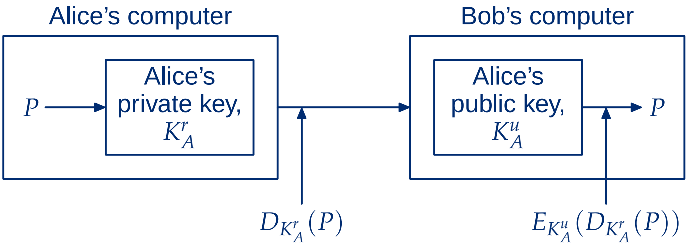
- The reciever must verify the claimed identity of a sender.
- The sender cannot deny having sent a message.
- The reciever cannot have created the message himself.

Symmetric key signatures



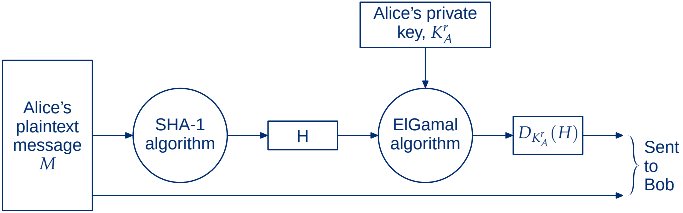
- Identity of A is proved to T by K_A.
- The fact of A's sending a message is proved by $K_T(A, P)$.
- B cannot forge having received a message from A because he does not know K_T.

Public key signatures



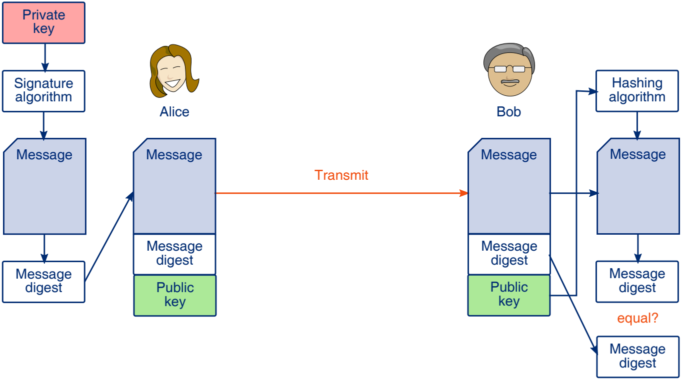
- Identity of A is proved by B's being able to encrypt a message with K_A^u.
- The fact of A's sending a message is proved by the existence of a message decrypted by K_A^r.
- B cannot forge having received a message from A because he cannot produce $D_{K_A^r}(M)$.

Digest signatures: DSS

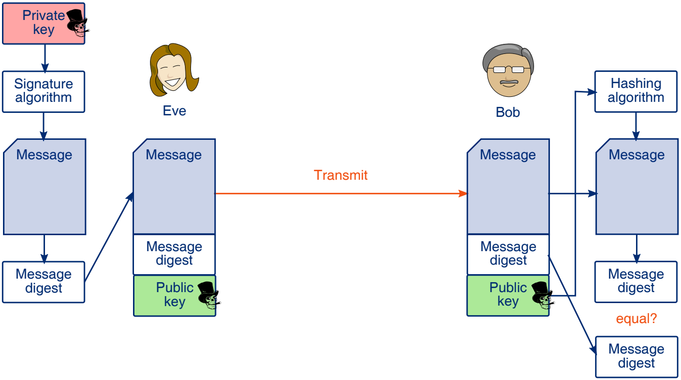


- For efficiency reasons, public-key decryption is applied to a short digest of the plaintext message.
- ElGamal public-key encryption/decryption algorithm is used.

Digest signature application



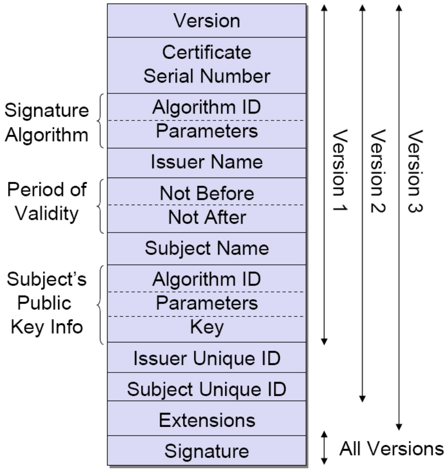
Digest signature attack



Public key certificates

- A certificate is a binding between an entity name and its public key.
- Certificates are issued by a “certification authority” (CA), a trusted third party.
- A certificate is generated locally on a computer.
- To grant a certificate its validity, a CA signes it with its private key.
- Since the CA’s public key is well known everybody can verify the validity of a certificate.

The structure of a X.509 certificate



Signature computation

The signature in the X.509 certificate is computed as:

$$CA << A >> = K^{-}_{CA}[V, SN, AI, CA, T_A, A, K^{+}_{A}],$$

where

V : X.509 version

SN : certificate serial number

AI : Algorithm ID

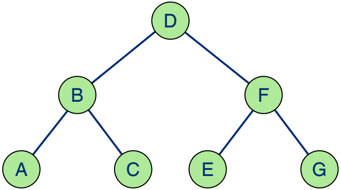
CA : certificate authority name

T_A : validity timespan of the certificate

A : subject name

K^{+}_{A} : subject public key

Certificate chains



- If a client trusts a root CA, it uses a top-down trust chain to verify a target certificate:

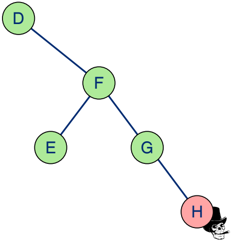
$$D << B >> \ B << C >>$$

- If a client trusts a local CA, e.g., G, is must traverse the certificate hierarchy:

$$G << F >> \ F << D >> \ D << B >> \ B << C >>$$

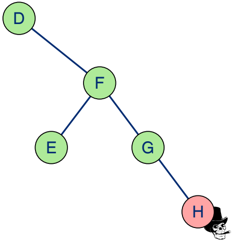
Rogue certificate authority

- An attacker obtains a legitimate (end) certificate G
- He creates a rogue certificate H and signs it with G as a CA



Rogue certificate authority

- An attacker obtains a legitimate (end) certificate G
- He creates a rogue certificate H and signs it with G as a CA
- The use of certificates as CA is restricted by the extension field 'basicConstraints'



Summary

- Integrity of data can be enforced by computing cryptografic hash functions (one-way, collision-resistant).
- Authentication and non-repudiation objectives are attained by digital signatures that combine public key cryptography with secure hashing.
- Binding of digital signatures to entities is achieved by putting the relevant information in X.509 certificate issued by a trusted certification authority (CA).