Elements of applied cryptography

Digital Signatures

- **Digital Signatures with appendix**
- **Digital signatures with message recovery**
- **Digital signatures based on RSA**

Informal properties

- **DEFINITION.** A digital signature is a number dependent on **some secret known only to the signer** and, additionally, on **the content of the message being signed**
- **PROPERTY.** A digital signature must be **verifiable**, i.e., if a dispute arises an **unbiased third party** must be able to solve the dispute **equitably, without requiring access to the signer's secret**

Classification

- **Digital signatures with appendix**
  - require the original message as input to the verification algorithm;
  - use hash functions
  - Examples: **ElGamal, DSA, DSS**, Schnorr
- **Digital signatures with message recovery**
  - do not require the original message as input to the verification algorithm;
  - the original message is recovered from the signature itself;
  - Examples: **RSA**, Rabin, Nyberg-Rueppel

Digital signatures with appendix

Definitions

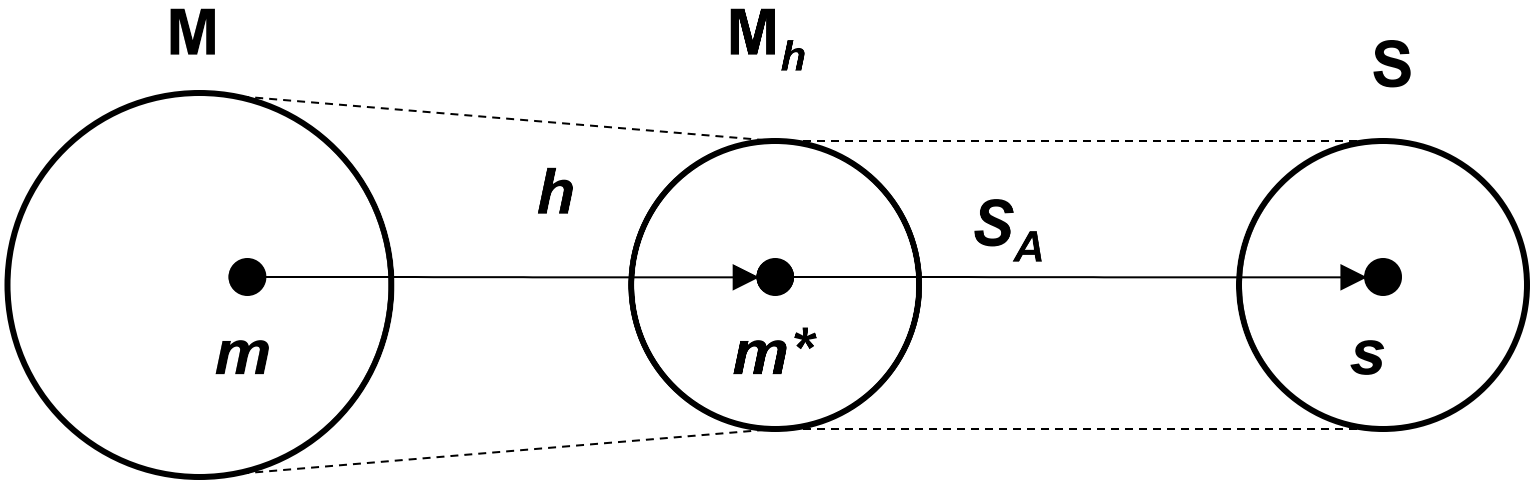
- *M* is the message space
- *h* is a hash function with domain *M*
- M_h is the image of *h*
- *S* is the signature space

Key generation

- Alice selects a private key which defines a ***signing algorithm S_A*** which is a **one-to-one** mapping $S_A: M_h \rightarrow S$
- Alice defines the corresponding public key defining the ***verification algorithm*** **V_A** such that **$V_A(m^*, s)$ = true** if **$S_A(m^*) = s$** and false otherwise, for all $m^* \in M_h$ and $s \in S$, where **$m^* = h(m)$** for $m \in M$.
- ***The public key V_A is constructed such that it may be computed without knowledge of the signer's private key S_A***

Digital signatures with appendix

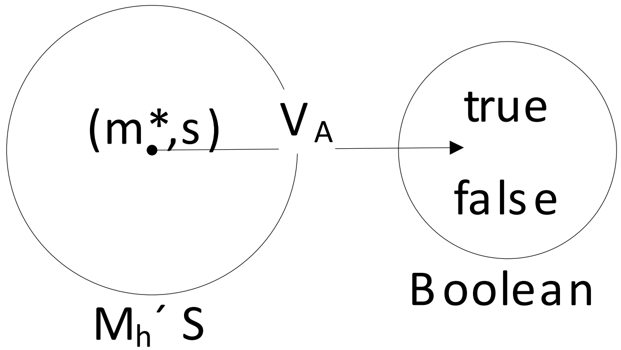
The signing process



Signature generation process

- Compute **m* = h(m)**, **s = S_A(m*)**
- Send **(m, s)**

Digital signatures with appendix



Signature verification process

- Obtain A’s public key $\mathbf{V_A}$
- Compute $\mathbf{m^* = h(m)}$, $\mathbf{u = V_A(m^*, s)}$
- Accept the signature iff **u = true**

Digital signatures with appendix

Properties of S_A and V_A

- (**efficiency**) S_A should be efficient to compute
- (**efficiency**) V_A should be efficient to compute
- (**security**) It should be **computationally infeasible** for an entity other than A to find an $m \in M$ and an $s \in S$ such that $V_A(m^*, s) = true$, where $m^* = h(m)$

Digital signature with message recovery

Definitions

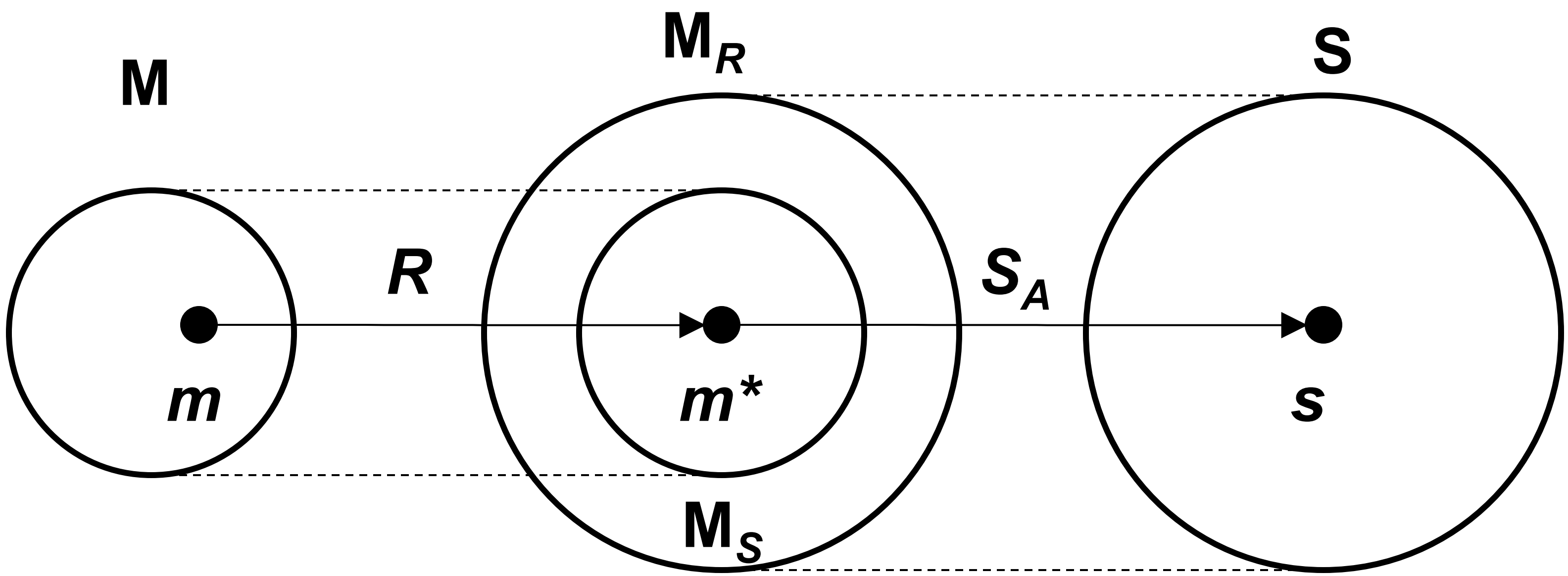
- **M** is the message space
- $\mathbf{M_S}$ is the signing space
- **S** is the signature space

Key generation

- **A** selects a private key defining a ***signing algorithm*** S_A which is a one-to-one mapping S_A: $M_S \rightarrow S$
- **A** defines the corresponding public key defining the ***verification algorithm*** V_A such that $V_A \bullet S_A$ is identity map on M_S.
- ***The public key V_A is constructed such that it may be computed without knowledge of the signer's private key S_A***

Digital signature with message recovery

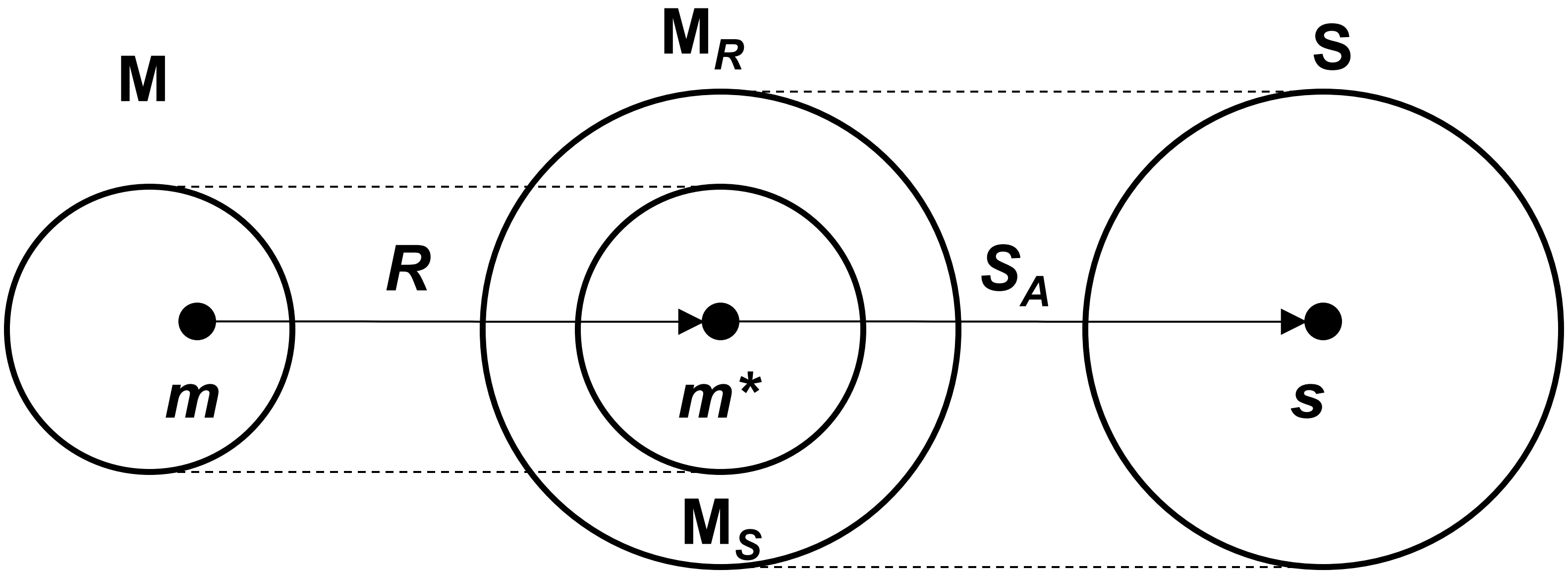
The signing process



- Compute ***m* = R*(*m*)**, ***R*** is a ***redundancy function*** (invertible)
- Compute $s = S_A(m^*)$

Digital signature with message recovery

The signing process



- Obtain authentic public key V_A
- Compute $m^* = V(s)$

► Verify if $m^* \in M_S$ (if not, reject the signature)

- Recover the message $m = R^{-1}(m^*)$

Digital signatures with message recovery

Properties of S_A and V_A

- (efficiency) S_A should be efficient to compute
- (efficiency) V_A should be efficient to compute
- (security) It should be **computationally infeasible** for an entity other than A to find an $s \in S$ such that $V_A(s) \in M_R$

Digital signatures with message recovery

The redundancy function

- R and R^{-1} are publicly known
- Selecting an appropriate R is *critical* to the security of the system

An example of bad redundancy function leading to existential forgery

- Let us suppose that $M_R \equiv M_S$
- R and S_A are bijections, therefore M and S have the same number of elements
- Therefore, for all $s \in S$, $V_A(s) \in M_R$. Therefore, it is "easy" to find an m for which s is the signature, $m = R^{-1}(V_A(s))$
- s is a valid signature for m (***existential forgery***)

Digital signatures with message recovery

A good redundancy function although too redundant

- Example
  - $M = \{m : m \in \{0, 1\}^n\}$, $M_S = \{m : m \in \{0, 1\}^{2n}\}$
  - $R: M \rightarrow M_S$, $R(m) = m||m$ (concatenation)
  - $M_R \subseteq M_S$
  - When n is large, $|M_R|/|M_S| = (1/2)^n$ is small. Therefore, for an adversary it is unlikely to choose an s that yields $V_A(s) \in M_R$
- **ISO/IEC 9776** is an international standard that defines a redundancy function for **RSA** and **Rabin**

Dig. sign. with appendix from message recovery

- **Signature generation**
  - Compute $\mathbf{m^* = \mathit{R}(\mathit{h}(\mathit{m}))}$, $\boldsymbol{s = S_A(m^*)}$
  - A's digital signature for ***m*** is ***s***

  ∀ $\langle \boldsymbol{m}, \boldsymbol{s} \rangle$ are made available to anyone who may wish to verify the signature
- **Signature verification**
  - Obtain A's public key $\boldsymbol{V_A}$
  - Compute $\boldsymbol{m}^* = \boldsymbol{R(h(m))}$, $\boldsymbol{m'} = \boldsymbol{V_A(s)}$, and $\boldsymbol{u = (m' == m^*)}$
  - Accept the signature iff ***u*** **= true**
- **Comment**
  - ***R*** is not security critical anymore and can be ***any*** *one-to-one mapping*

Types of attacks

BREAKING A SIGNATURE

1. **Total break** – adversary is able to compute the signer's private key
2. **Selective forgery** – adversary controls the messages whose signature is forged
3. **Existential forgery** – adversary has no control on the messages whose signature is forged

Types of attacks

BASIC ATTACKS

- **KEY-ONLY ATTACKS** – adversary knows only the signer's public key
- **MESSAGE ATTACKS**

a. **known-message attack**

An adversary has signatures for a set of messages which are known by the adversary but not chosen by him

a. **chosen-message attack**

In this case messages are chosen by the adversary

b. **adaptive chosen-message attack**

In this case messages are adaptively chosen by the adversary

Attacks: considerations

- **Adaptive chosen-message attack**
  - It is the most difficult attack to prevent
  - Although an adaptive chosen-message attack may be infeasible to mount in practice, a well-designed signature scheme should nonetheless be designed to protect against the possibility
- **The level of security may vary according to the application**
  - **Example 1.** When an adversary is only capable of mounting a key-only attack, it may suffice to design the scheme to prevent the adversary from being successful at selective forgery.
  - **Example 2**. When the adversary is capable of a message attack, it is likely necessary to guard against the possibility of existential forgery.

Attacks: considerations

- ***Hash functions and digital signature processes***
  - When a hash function ***h*** is used in a digital signature scheme (as is often the case), ***h* should be a fixed part of the signature process** so that an adversary is unable to take a valid signature, replace ***h*** with a weak hash function, and then mount a selective forgery attack.
  - Example. Let $\langle \boldsymbol{m}, \boldsymbol{s} \rangle$ where $\boldsymbol{s} = \boldsymbol{S_A(h(m))}$.
    Let adversary be able to replace ***h*** with a weaker hash function ***g*** that is vulnerable to selective forgery.
    Then the adversary can

    1. determine $\boldsymbol{m'}$ such that $\boldsymbol{g(m') = h(m)}$; and
    2. replace ***m*** with $\boldsymbol{m'}$

Digital signatures based on RSA

Introductory comments

- Since the encryption transformation is a bijection, digital signatures can be created by reversing the roles of encryption and decryption
- Digital signature with ***message recovery***
- $M_S \equiv S \equiv V_n$
- A redundancy function $R: M \rightarrow V_n$ is chosen and is public knowledge

Key generation

1. Generate two **large**, **distinct primes** p, q (100÷200 decimal digits)
2. Compute $n = p \times q$ and $\phi = (p-1) \times (q-1)$
3. Select a **random number** $1 < e < \phi$ such that $\gcd(e, \phi) = 1$
4. Compute the **unique** integer $1 < d < \phi$ such that $ed \equiv 1 \bmod \phi$
5. (d, n) is the private key
6. (e, n) is the public key

At the end of key generation, p and q must be destroyed

Signature generation and verification

Signature generation. In order to sign a message m, A does the following

1. Compute $m^* = R(m)$ an integer in $[0, n–1]$
2. Compute $s = m^{*d} \bmod n$
3. A's signature for m is s

Signature verification. In order to verify A's signature s and recover message m, B does the following

1. Obtain A's authentic public key (e, n)
2. Compute $m^* = s^e \bmod n$
3. Verify that m^* is in M_R; if not reject the signature
4. Recover $m = R^{-1}(m^*)$

Proof that verification works

- **Theorem**. If s is a signature for a message m, then $s = m^{*d} \bmod n$ where $m^* = R(m)$.
- ***Proof***.
  - Since $ed = 1 \pmod{\phi}$, $s^e = m^{*ed} = m^* \pmod{n}$. Finally, $R^{-1}(m^*) = R^{-1}(R(m)) = m$.

Possible attacks

- **Integer factorization**
  - Factorization of ***n*** lead to total break.
  - A should choose ***p*** and ***q*** so that factoring ***n*** is a computationally infeasible task
- **Multiplicative property of RSA: requirement on R**
  - A necessary condition for avoiding existential forgery is that ***R*** must not satisfy the multiplicative property.

RSA signature in practice

Reblocking problem. If Alice wants to send Bob a secret and signed message to Bob then it must be $n_A < n_B$

- There are various ways to solve the problem
  - **reordering**: the operation with the smaller modulus is performed first; *however the preferred order is always to sign first and encrypt later*
  - **two moduli for entity**: each entity has two moduli; moduli for signing (e.g., t-bits) are always smaller of all possible moduli for encryption (e.g., t+1-bits)
  - **ad-hoc format of the moduli**

RSA signature in practice

- *Redundancy function*
  - A suitable redundancy function is necessary in order to avoid existential forgery
  - IOS/IEC 9796 (1991) defines a mapping that takes a k-bit integer and maps it into a 2k-bits integer
- *The RSA digital signature scheme with appendix*
  - MD5 (128 bit)
  - PKCS#1 specifies a redundancy function mapping 128-bit integer to a k-bit integer, where k is the modulus size ($k \geq 512$, $k = 768, 1024$)

RSA signature in practice

- *Performance characteristics*
  - Let $|p| = |q| = k$ then
  - **signature generation requires $O(k^3)$ bit operations**
  - **signature verification, in the case of small public exponent, requires $O(k^2)$ bit operations**
  - Suggested value for **e** in practice are 3 and $2^{16}+1$. Of course, p and q must be chosen so that $\gcd(e, (p-1)(q-1)) = 1$.
  - **The RSA signature scheme is ideally suited to situations where signature verification is the predominant operation being performed**.
    - Example. A trusted third party creates a public-key certificate for an entity A. This requires only one signature generation, and this signature may be verified many times by various other entities

RSA signature in practice

- *Parameter selection*
  - bitsize of the modulus: miminum 768; at least 1024 for signatures of longer lifetime or critical for overall security of a large network (i.e., the private key of a certification authority)
  - No weaknesses have been reported when the public exponent e is chosen to be a small number such as 3 or $2^{16}+1$.
  - It is not recommended to restrict the size of the private exponent d in order to improve the efficiency of signature generation
- *Bandwidth efficiency*
  - By definition, BWE = log2 ($|M_S|$) / log2 ($|M_R|$)
  - For (RSA, ISO/IEC 9796), BWE = 0.5, that is, with a 1024-bits modulus can be signed 512-bits messages

RSA signature in practice

- System wide parameters
  - Each entity must have a distinct RSA modulus; it is insecure to use a system-wide modulus
  - The public exponent e can be a system-wide parameter, and is in many applications. In this case, the low exponent attack must be considered
- Short vs. long messages
  - Suppose n is a 2k-bit RSA modulus which is used to sign k-bit messages (i.e., BWE is 0.5)
  - Suppose entity A wishes to sign a kt-bit message m
  - For t = 1 RSA with message recovery is more efficient;
  - For t > 1, RSA with appendix is more efficient

RSA, hash functions and forgery

- Digital signature and preimage resistance
  - Go to here.

DIGITAL SIGNATURES BASED ON ELGAMAL

ElGamal's digital signature

Discrete Logarithm Systems

- Let p be a prime, q a prime divisor of p–1 and $g \in [1, p-1]$ has order q
- Let x be the *private key* selected at random from $[1, q-1]$
- Let y be the corresponding *public key* $y = g^x \bmod p$

Discrete Logarithm Problem (DLP)

- Given (p, q, g) and y, determine x

ElGamal's digital signature

- **Signature**
  - select $k \in Z_{p-1}^*$ randomly
  - $r = g^k \bmod p$, $s = (h(m) - xr)k^{-1} \bmod (p-1)$
  - The pair (r, s) is the digital signature for m
- **Verification**
  - Verify that $1 \leq r \leq p-1$; if not reject the signature
  - Compute $v_1 = y^r r^s \bmod p$
  - Compute $h(m)$ and $v_2 = g^{h(m)} \bmod p$
  - Accept the signature only if $v_1 = v_2$.

ElGamal's digital signature

Proof

- If the digital signature (r, s) has been produced by Alice then s = $(h(m)-xr)k^{-1}$ mod (p–1).
- Multiplying both sides by k gives $ks = (h(m)-xr)$ mod (p–1). Rearranging yields $h(m) \equiv ks+xr$ mod (p–1).
- This implies that $g^{h(m)} \equiv g^{ar+ks} \equiv (g^x)^r r^s$ mod p
- Thus $v_1 = v_2$ as required.

ElGamal's digital signature

Security

- In order to forge a signature, an adversary can select ***k*** at random, compute $r = g^k \bmod p$. Than he has to compute $s = (h(m) – xr)k^{–1} \bmod (p–1)$. If the DLP is computationally infeasible, the adversary can do no better than to choose an s at random; the success probability is $1/p$ which is negligible for large p.
- A different k must be selected for different messages otherwise the secret key x can be revealed
- If no hash function h is used, an adversary can easily mount an existential forgery attack.
- If the check on r is not done, an adversary can sign messages of its choice provided it has one valid signature produced by Alice

AUTHENTICATION VS NON-REPUDIATION

Non-repudiation

- Non-repudiation prevents a signer from signing a document and subsequently being able to successfully deny having done so.
- ***Non-repudiation vs authentication of origin***
  - Authentication (based on symmetric cyptography) allows a party to convince **itself** or a **mutually trusted party** of the integrity/authenticity of a given message at a given time t_0
  - Non-repudiation (based on public-key cyptography) allows a party to convince **others** at any time $t_1 \geq t_0$ of the integrity/authenticity of a given message at time t_0

Alice's digital signature for a given message depends on the message and a secret **known to Alice only (the private key)**

Non-repudiation

- Data origin authentication as provided by a digital signature is valid only while the *secrecy* of the signer's *private key* is maintained
- A threat that must be addressed is a signer who *intentionally* discloses his private key, and thereafter claims that a *previously* valid signature was forged
- This threat may be addressed by
  - ***preventing direct access to the key***
  - ***use of a trusted timestamp agent***
  - ***use of a trusted notary agent***

Thanks for attention!