Digital Signatures

Gaspare FERRARO
CyberSecNatLab
Matteo ROSSI
Politecnico di Torino

CYBER CHALLENGE.IT

CYBERSECURITY NATIONAL LABORATORY

https://cybersecnatlab.it

License & Disclaimer

License Information

This presentation is licensed under the Creative Commons BY-NC License

To view a copy of the license, visit:

http://creativecommons.org/licenses/by-nc/3.0/legalcode

Disclaimer

- We disclaim any warranties or representations as to the accuracy or completeness of this material.
- Materials are provided “as is” without warranty of any kind, either express or implied, including without limitation, warranties of merchantability, fitness for a particular purpose, and non-infringement.
- Under no circumstances shall we be liable for any loss, damage, liability or expense incurred or suffered which is claimed to have resulted from use of this material.

Goal

- Give the definition and show usage of digital signatures
- Show the differences between hash, MAC, and digital signatures
- Learn how to perform digital signatures with RSA
- Introduce the DSA algorithm and its weaknesses

Prerequisites

- Lectures:
  - *CR_0.1 - Number Theory and modular arithmetic*
  - *CR_1.1 - Introduction to cryptography*
  - *CR_2 - Public-key cryptography*
  - *CR_3.1 - Hash Functions*

Outline

- Introduction
- Digital Signatures from RSA
- The Digital Signature Algorithm
- Nonce reuse in DSA

Outline

- Introduction
- Digital Signatures from RSA
- The Digital Signature Algorithm
- Nonce reuse in DSA

Introduction

- Recap from lecture CR_3.1:
  - Message integrity: can the recipient be confident that the message has not been accidentally modified?
  - Authentication: can the recipient be confident that the message originated from the sender?

Introduction

- A step forward non-repudiation:
  - Protection against an individual falsely denying having performed a particular action
  - Provides the capability to determine whether a given individual took a particular action such as creating information, sending a message, approving information, and receiving a message

Introduction

Informally: a digital signature is like a MAC, but with public key cryptography

↓

No need of a shared secret key

Hash vs MAC vs Signatures

Primitive	Integrity	Authentication	Non-repudiation
Hash	Yes	No	No
MAC	Yes	Yes	No
Digital Signature	Yes	Yes	Yes

Signatures vs MACs

- Pros of digital signatures
  - No need of sharing a key
  - Non-repudiation property

- Cons of digital signatures
  - Slow compared to MACs

Signatures in practice

- In practice, a digital signature is a pair of functions, *Sign* and *Verify*, such that
  - *Sign* takes *an hash* of a message of arbitrary length and a key and produces a fixed-length string, called *signature*
  - *Verify* takes *the hash* of the message, the key and the signature, and outputs true if the signature is valid and false otherwise

Why hashes?

- Hashing is useful to avoid to have:
  - too short messages
  - have messages longer than the modulus used in the sign and verify functions
- Recall the vulnerabilities presented in the lecture CR_2.3 - Attacks on RSA

Outline

- Introduction
- Digital Signatures from RSA
- The Digital Signature Algorithm
- Nonce reuse in DSA

Signing with RSA

- A basic signature scheme using RSA can be constructed as follows:
  - The Sign function for a message m is
  $$s = m^d \bmod n$$
  - The Verify function is
  $$m = s^e \bmod n$$
- Issues?

Forgery

- Some signatures are independent from the value of d:
  - The signature of 0 is always 0
  - The signature of 1 is always 1
  - The signature of n-1 is always n-1

Blinding

- Using the homomorphic properties of RSA, we can sign an arbitrary message M without asking directly to the oracle to sign it:
  - Select a value R
  - Ask to sign $(\mathrm{R^eM}) \rightarrow Sign(\mathrm{R^eM}) = (\mathrm{R^eM})^d = \mathrm{RM^d} \bmod \mathrm{n}$
  - Use the multiplicative inverse of R to get a signature for M from the signature of $(\mathrm{R}^e\mathrm{M})$:
    - $Sign(M) = Sign(R^eM)R^{-1} = (RM^d)R^{-1} = M^d \; mod \; n$

Outline

- Introduction
- Digital Signatures from RSA
- The Digital Signature Algorithm
- Nonce reuse in DSA

History

- In 1982, the US government asked for proposals for digital signature standards
- In 1991, the Digital Signature Algorithm (DSA) was proposed by NIST and standardized
- Since 2019, DSA is no longer recommended by NIST, and it has been mostly replaced by its elliptic curve-based equivalent algorithm (ECDSA)

Overview

- DSA is based on 4 algorithms:
  - Parameters generation
  - Key generation
  - Sign algorithm
  - Verify algorithm

Parameters generation

- Pick a cryptographic hash function H (usually SHA1)
- Pick a prime number q
- Pick a prime number p such that $p-1$ is multiple of q
- Pick a number h in $\{2,3,\ldots,p-2\}$ (usually $h=2$) and $g = h^{(p-1)/q} \bmod p$
- The values (H,p,q,g) are the (publicly shared) parameters of the DSA instance

Key generation

- Each user generates a key as follows:
  - Pick x in $\{1,2,\ldots,q-1\}$
  - Set $y = g^x \bmod p$
  - x is the private key, y is the public one

Signing

- A signature of a message m is made as follows:
  - Pick a random value k (called the *nonce*) in $\{1, \ldots, q-1\}$
  - Compute $r = (g^k \bmod p) \bmod q$
  - Compute $s = (k^{-1}(H(m) + xr)) \bmod q$
  - The pair (r, s) is the signature of m

Verifying

- Given a signature (r, s) and a message m, the verification is made as follows:
  - Compute $u_1 = H(m)s^{-1} \bmod q$
  - Compute $u_2 = rs^{-1} \bmod q$
  - $v = (g^{u_1}y^{u_2}\, mod\, p) \bmod \mathrm{q}$
- The signature is valid if and only if $v = r$

Outline

- Introduction
- Digital Signatures from RSA
- The Digital Signature Algorithm
- Nonce reuse in DSA

Nonce reuse

- The main problem for DSA stems from the choice of the nonce k
- In the next slide, we show just what happens if k is used more than once
- In general, using not random (or biased) nonces is a bad idea

Nonce reuse

- Suppose to have two messages m_1, m_2 signed by the same user with the same nonce k
- Let's call the signatures (r_1, s_1) and (r_2, s_2)
- We can simply recover the private key x as follow:
  - $x = \left(s_2 H(m_1) - s_1 H(m_2)\right)(r_2 s_1 - r_1 s_2)^{-1} \bmod q$

Gaspare FERRARO

CyberSecNatLab

Matteo ROSSI

Politecnico di Torino

Digital Signatures

CYBER CHALLENGE.IT

CYBERSECURITY NATIONAL LABORATORY

https://cybersecnatlab.it