

Cryptography V: Digital Signatures

Computer Security Lecture 6

David Aspinall

School of Informatics
University of Edinburgh

31st January 2013

Outline

- Basics
- Constructing signature schemes
- Security of signature schemes
- ElGamal
- DSA
- Summary

Aims

- ▶ **Digital signatures** allow a principal to cryptographically bind (a representation of) its identity to a piece of information.
- ▶ Signatures can help establish security properties such as:
 - ▶ authentication
 - ▶ accountability/non-repudiation
 - ▶ unforgeability
 - ▶ integrity
 - ▶ verifiability by independent, public or 3rd party
- ▶ Digital signatures are the asymmetric analogue of MACs, with a crucial difference. MACs can't distinguish which of A or B provided integrity to a message (so no non-repudiation or independent verifiability).
- ▶ NB: **electronic signature** is a more general notion.

Handwritten versus Digital Signatures



- | | |
|--------------------------|---------------------------------|
| ink binds to paper | cryptographically bound to data |
| verifier needs signature | verifier needs public key |
| signatures always same | depends on document |
| copies apparent | copies indistinguishable |
| signer saw document | computer added signature |
| have legal impact | may have legal impact |

Signature mechanism

A signature mechanism for principal A is given by:

- ▶ A message space \mathcal{M} of messages for signing
- ▶ A set \mathcal{S} of *signatures* (e.g. strings $\{0, 1\}^n$)
- ▶ A secret **signing function** $S_A : \mathcal{M} \rightarrow \mathcal{S}$
- ▶ A public **verification function** $V_A : \mathcal{M} \times \mathcal{S} \rightarrow \text{Bool}$

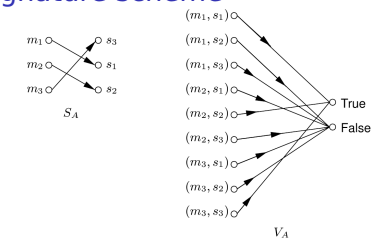
satisfying the correctness and security properties:

1. $V_A(m, s) = \text{true}$ if and only if $S_A(m) = s$.
2. For any principal other than A, it is computationally infeasible to find for any $m \in \mathcal{M}$, an $s \in \mathcal{S}$ such that $V_A(m, s) = \text{true}$.

Usually use a public algorithm yielding key-indexed families $\{S_s \mid s \in \mathcal{K}\}$ of signing and verification functions $\{V_v \mid v \in \mathcal{K}\}$. Principal advertises v .

Remark: nobody has proved a signature mechanism satisfying 2 exists, although there are good candidates.

Using a signature scheme



- ▶ To sign a message the *signer* A
 1. Computes $s = S_A(m)$.
 2. Sends the pair (m, s) .
- ▶ To verify that a signature s on a message m was created by A, another principal, the *verifier*:
 1. Obtains the verification function V_A for A.
 2. Computes $u = V_A(m, s)$
 3. **Accepts** the signature if $u = \text{true}$, **Rejects** it if $u = \text{false}$.

Digital signatures with a TTP

- ▶ Given a trusted third party, it is possible to use symmetric cryptography techniques.
- ▶ Let secure Sam S be the TTP, who shares a key with each principal.
- ▶ For A to send a signed contract M to B , S acts as an intermediary.

Message 1. $A \rightarrow S: \{M\}_{K_{AS}}$

Message 2. $S \rightarrow B: \{M\}_{K_{BS}}$

(like Wide Mouthed Frog key exchange protocol, M should include time-stamps and names).

- ▶ If A and B disagree about a signature, a judge J can verify the contracts also using S :

Message 1. $J \rightarrow S: \{M\}_{K_{AS}}, \{M\}_{K_{BS}}$

Message 2. $S \rightarrow J: \{yes\ or\ no\}_{K_{JS}}$

Digital signatures from PK encryption

- ▶ Suppose we have a public-key encryption scheme with $\mathcal{M} = \mathcal{C}$, and (d, e) a key-pair. Then because E_e and D_d are both permutations on \mathcal{M} , we have that:

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

A public-key scheme of this type is called *reversible*.

- ▶ RSA is reversible, but not every PK scheme is.
- ▶ We can define a digital signature scheme by reversing encryption and decryption:
 - ▶ Message space \mathcal{M} , signature space $\mathcal{C} (= \mathcal{M})$.
 - ▶ the signing function $S_A = D_d$
 - ▶ the verification function V_A is defined by

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

Attacks on signature schemes [HAC]

- ▶ An adversary wants to forge signatures. Cases:
 1. **Total break.** Adversary can compute the private key or find an equivalent signing function.
 2. **Selective forgery.** Adversary can create a valid signature for some chosen message, without using the signer.
 3. **Existential forgery.** Adversary can create a valid signature for at least one message, without explicit choice of the message. May involve signer.
- ▶ The adversary may have different knowledge levels. For PK schemes:
 1. **Key-only attack:** adversary only knows PK.
 2. **Known-message attack:** adversary has signatures for some known (not chosen) messages.
 3. **Chosen-message attack:** adversary can obtain signatures for messages of his choosing. Messages may be determined in advance or in **adaptive** way, using signer as oracle.

Existential forgery

- ▶ The previous scheme is too simple because signatures are forgeable: a principal B can generate a random $s \in \mathcal{S}$ as a signature, apply the public encryption function to get a message $m = E_e(s)$, and transmit (m, s) .
- ▶ Obviously this verifies! It is an example of **existential forgery**.
- ▶ The message m is not likely to be of B 's choosing (and probably garbage).
- ▶ But this ability violates property 2 given earlier.

Signatures with redundancy

- ▶ A fix to reduce likelihood of existential forgery is to take $\mathcal{M}' \subset \mathcal{M}$ to be messages with a special *redundant* structure, which is publicly known e.g., messages padded to an even length, surrounded with a fixed bit pattern.
- ▶ This format is easily recognized by the verifier:

$$V_A(s) = \begin{cases} \text{true} & \text{if } E_e(s) \in \mathcal{M}', \\ \text{false} & \text{otherwise.} \end{cases}$$

- ▶ Now A only transmits the signature s , since the message $m = E_e(s)$ can be recovered by the verification function.
- ▶ This property is **message recovery**, the scheme is called a **signature scheme with recovery**.
- ▶ Existential forgery is now less likely.

Signatures and hash functions

- ▶ In practice, usually the signing function is constructed by first making a hash of the input document, and signing that. Reasons:
 1. efficiency: signature is on smaller text
 2. avoid attacks on cipher system
- ▶ Signer: computes and transmits (m, s) where $s = S_A(h(m))$.
- ▶ Verifier: computes $h(m)$ and verifies $V_A(h(m), s)$.
- ▶ The hash function must satisfy appropriate properties (see *Hash Functions* lecture).
- ▶ This is called a **signature scheme with appendix**.

RSA Signatures

- ▶ Setup: $n = pq$ computed as product of two primes. $ed \equiv 1 \pmod{\phi(n)}$. (e, n) is the public key.
- ▶ To sign a message m , compute the signature $s = h(m)^d \pmod{n}$. Only the owner of the private key d is able to compute the signature.
- ▶ To verify the signature, upon receipt of (m, s) , compute $s^e \pmod{n}$ and verify whether it equals $h(m)$

Distributed RSA Signatures

- ▶ Signatures can optionally be *distributed* so that each of t users contributes to the signature. A trusted party T computes t shares such that

$$d = \sum_{i=1}^t d_i \pmod{\phi(n)}$$

- and securely distributes d_i to each user i .
- ▶ To compute a signature on a message m , each user i computes $o_i = h(m)^{d_i} \pmod{n}$.
- ▶ A signer can compute the resultant signature as

$$s = \prod_{i=1}^t o_i \pmod{n}$$

- ▶ *Secret sharing* can also be used so that $l < t$ users could be used to construct a signature.

ElGamal signatures

- ▶ Setup as encryption: p an appropriate prime, g a generator of \mathbf{Z}_p^* , and the private signing key, d a random integer with $1 \leq d \leq p - 2$.
- ▶ The public verification key is $(p, g, g^d \pmod{p})$.
- ▶ To sign a message m , $0 \leq m \leq p$, the signer picks a random secret number r with $1 \leq r \leq p - 2$ and $\gcd(r, p - 1) = 1$, and computes:

$$\mathbf{S}_d(m) = (e, s) \quad \text{where } e = g^r \pmod{p} \\ de + rs \equiv m \pmod{p - 1}.$$

- ▶ The verification function checks that $1 \leq e \leq p - 1$, and an equation:

$$\mathbf{V}_{(p, g, g^d)}(m, (e, s)) = \begin{cases} \text{true} & \text{if } (g^d)^e e^s \equiv g^m \pmod{p}, \\ \text{false} & \text{otherwise.} \end{cases}$$

- ▶ Verification works because for a correct signature,

$$(g^d)^e e^s \equiv g^{de+rs} \equiv g^m \pmod{p}.$$

From ElGamal to DSA

- ▶ The Digital Signature Algorithm is part of the NIST *Digital Signature Standard* [FIPS-186].
- ▶ Based on ElGamal, but with improved efficiency.
- ▶ The first digital signature scheme to be recognized by any government.
- ▶ Based on two primes: p , which is 512–1024 bits long, and q , which is a 160-bit prime factor of $p - 1$. A signature signs a SHA-1 hash value of a message. (In fact, ElGamal signing should be used with a hash function to prevent existential forgery)
- ▶ **Security** of both ElGamal and DSA schemes relies on the intractability of the DLP.
- ▶ Comparison with RSA signature scheme: key generation is faster; signature generation is about the same; DSA verification is slower. Verification is the most common operation in general.

Summary: Digital Signature Schemes

- ▶ **RSA, ElGamal, DSA** already described. There are several variants of ElGamal, including schemes with message recovery.
- ▶ Notice difference between *randomized* and *deterministic* schemes.
- ▶ Schemes for **one-time signatures** (e.g., Rabin, Merkle), require a fresh public key for each use.
 - ▶ Typically more efficient than RSA/ElGamal methods.
 - ▶ But tedious for multiple documents
- ▶ E-cash protocols use **blind signature** schemes that prevent the signer (e.g., a bank) linking a signed message (e.g., the cash) with the user.
- ▶ For real world security guarantees:
 - ▶ **obtaining correct public key** is vital;
 - ▶ non-repudiation supposes that **private key has not been stolen**;
 - ▶ we may require **secure time stamps**.

References

- ▶ Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone, editors. *Handbook of Applied Cryptography*. CRC Press Series on Discrete Mathematics and Its Applications. CRC Press, 1997.
Online version at <http://www.cacr.math.uwaterloo.ca/hac>.
Digital signatures covered in Section 1.6 and Chapter 11.
- ▶ Nigel Smart. *Cryptography: An Introduction*. McGraw-Hill, 2003. Third edition online: http://www.cs.bris.ac.uk/~nigel/Crypto_Book/

Recommended Reading

Chapter 14 (14.2–14.4, 14.7) of Smart (3rd Ed).