CPSC 367: Cryptography and Security

Michael J. Fischer

Lecture 12
February 26, 2019

Asymmetric Digital Signatures

Asymmetric digital signatures

An asymmetric (public-key) digital signature can be viewed as a 2-key MAC, just as an asymmetric (public-key) cryptosystem is a 2-key version of a classical cryptosystem.

In the literature, the term *digital signature* generally refers to the asymmetric version.

Asymmetric digital signatures

Let $\mathcal{M}$ be a *message space* and $\mathcal{S}$ a *signature space*.

A *signature scheme* consists of a private *signing key* d, a public *verification key* e, a *signature function* $S_d : \mathcal{M} \to \mathcal{S}$, and a *verification predicate* $V_e \subseteq \mathcal{M} \times \mathcal{S}$.[1]

A *signed message* is a pair $(m, s) \in \mathcal{M} \times \mathcal{S}$. A signed message is *valid* if $V_e(m, s)$ holds, and we say that (m, s) is *signed with respect to* e.

[1] As with RSA, we denote the private component of the key pair by the letter d and the public component by the letter e, although they no longer have same mnemonic significance.

Fundamental property of a signature scheme

Basic requirement:

The signing function always produces a valid signature, that is,

$$V_e(m, S_d(m)) \tag{1}$$

holds for all $m \in \mathcal{M}$.

Assuming e is Alice's public verification key, and only Alice knows the corresponding signing key d, then a signed message (m, s) that is valid under e identifies Alice with m (possibly erroneously, as we shall see).

What does a digital signature imply?

We like to think of a digital signature as a digital analog to a conventional signature.

- A conventional signature binds a person to a document. Barring forgery, a valid signature indicates that a particular individual performed the action of signing the document.
- A digital signature binds a signing key to a document. Barring forgery, a valid digital signature indicates that a particular signing key was used to sign the document.

However, there is an important difference. A digital signature only binds the signing key to the document.

Other considerations must be used to bind the individual to the signing key.

Disavowal

An individual can always disavow a signature on the grounds that the private signing key has become compromised.

Here are two ways that this can happen.

- Her signing key might be copied, perhaps by keystroke monitors or other forms of spyware that might have infected her computer, or a stick memory or laptop containing the key might be stolen.
- She might deliberately publish her signing key in order to relinquish responsibility for documents signed by it.

For both of these reasons, one cannot conclude without a reasonable doubt that a digitally signed document was indeed signed by the purported holder of the signing key.

Practical usefulness of digital signatures

This isn't to say that digital signatures aren't useful; only that they have significantly different properties than conventional signatures.

In particular, they are subject to disavowal by the signer in a way that conventional signatures are not.

Nevertheless, they are still very useful in situations where disavowal is not a problem.

Digital Signature Algorithms

RSA digital signature scheme

Let n be an RSA modulus and (e, d) an RSA key pair.
e is public and d is private as usual.

- Signing function: $S_d(m) = D_d(m)$
- Verification predicate: $V_e(m, s) \Leftrightarrow m = E_e(s)$.

Must verify that $V_e(m, S_d(m))$ holds for all messages m, i.e., that $m = E_e(D_d(m))$ holds.

This is the reverse of the requirement for RSA to be a valid cryptosystem, viz. $m = D_d(E_e(m))$ for all $m \in \mathbf{Z}_m$.

RSA satisfies both conditions since

$$m \equiv D_d(E_e(m)) \equiv (m^e)^d \equiv (m^d)^e \equiv E_e(D_d(m)) \pmod{n}.$$

Commutative cryptosystems

A cryptosystem with this property that $D_d \circ E_e = E_e \circ D_d$ is said to be *commutative*, where "$\circ$" denotes functional composition.

Indeed, any commutative public key cryptosystem can be used for digital signatures in exactly this same way as we did for RSA.

Signatures from non-commutative cryptosystems

What if E_e and D_d do not commute?

One could define:

- Signing function: $S_e(m) = E_e(m)$
- Verification predicate: $V_d(m, s) \Leftrightarrow m = D_d(s)$.

Every validly-signed message $(m, S_e(m))$ would verify since $D_d(E_e(m)) = m$ is the basic property of a cryptosystem.

Now, Alice has to keep e private and make d public, which she can do. However, the resulting system might not be secure, since even though it may be hard for Eve to find d from e and n, it does not follow that it is hard to find e from d and n.

Interchanging public and private keys

For RSA, it is just as hard to find e from d as it is to find d from e.

That's because RSA is completely symmetric in e and d.

Not all cryptosystems enjoy this symmetry property.

ElGamal cryptosystem is not symmetric

The ElGamal scheme discussed in lecture 11 is based on the equation

$$b = g^y \pmod{p},$$

where y is private and b public.

Finding y from b, g, p is the discrete log problem — believed to be hard.

Finding b from y, g, p, is straightforward, so the roles of public and private key cannot be interchanged while preserving security.

ElGamal found a different way to use the ideas of discrete logarithm to build a signature scheme, which we discuss later.

Security of Digital Signatures

Desired security properties of digital signatures

Digital signatures must be difficult to forge.

Some increasingly stringent notions of forgery-resistance:

- Resistance to forging valid signature for particular message m.
- Above, but where adversary knows a set of valid signed messages $(m_1, s_1), \ldots, (m_k, s_k)$, and $m \notin \{m_1, \ldots, m_k\}$.
- Above, but where adversary can choose a set of valid signed messages, specifying either the messages (corresponding to a chosen plaintext attack) or the signatures (corresponding to chosen ciphertext attack).
- Any of the above, but where one wishes to protect against generating any valid signed message (m', s') different from those already seen, not just for a particular predetermined m. This is called *existential forgery*.

Forging random RSA signed messages

RSA signatures are indeed vulnerable to existential forgery.

An attacker simply chooses s' at random and computes $m' = E_e(s')$.

The signed message (m', s') is trivially valid since the verification predicate is simply $m' = E_e(s')$.

Importance of random signed messages

One often wants to sign random strings.

For example, in the Diffie-Hellman key exchange protocol discussed in lecture 10, Alice and Bob exchange random-looking numbers $a = g^x \bmod p$ and $b = g^y \bmod p$.

In order to discourage man-in-the-middle attacks, they may wish to sign these strings (assuming they already have each other's public verification keys).

With RSA signatures, Mallory could feed bogus signed values to Alice and Bob. The signatures would check, and both would think they had successfully established a shared key k when in fact they had not.

Using Digital Signatures

Adding redundancy

Recall: RSA signatures are subject to existential forgery.

Redundancy can be used to prevent this.

Example: Prefix a fixed string σ to the front of each message before signing.

This gives rise to a variant RSA signature scheme (S_d^σ, V_e^σ).

- Signing function: $S_d^\sigma(m) = D_d(\sigma m)$
- Verification predicate: $V_e^\sigma(m, s) \iff \sigma m = E_e(s)$.

Security of signatures with fixed redundancy

The security of this scheme depends on the mixing properties of the encryption and decryption functions, that is, the extent to which each output bit depends on most of the input bits.

Not all cryptosystems mix well.

For example, a block cipher used in ECB mode (see lecture 6 and lecture 8) encrypts a block at a time, so each block of output bits depends only on the corresponding block of input bits.

Forging signatures with fixed redundancy

Suppose it happens that

$$S_d^{\sigma}(m) = D_d(\sigma m) = D_d(\sigma) \cdot D_d(m).$$

Then Mallory can forge random messages assuming he knows just one valid signed message (m_0, s_0). Here's how.

- He knows that $s_0 = D_d(\sigma) \cdot D_d(m)$, so from s_0 he extracts the prefix $s_{00} = D_d(\sigma)$.
- He now chooses a random s'_{01} and computes $m' = E_e(s'_{01})$ and $s' = s_{00} \cdot s'_{01}$.
- The signed message (m', s') is valid since $E_e(s') = E_e(s_{00} \cdot s'_{01}) = E_e(s_{00}) \cdot E_e(s'_{01}) = \sigma m'$.

Signing Message Digests

Message digests

A better way to prevent forgery is to sign a *message digest* of the message rather than sign m itself.

A message digest function h, also called a *cryptographic one-way hash function* or a *fingerprint function*, maps long strings to short random-looking strings.

- To sign a message m, Alice computes $S_d(m) = D_d(h(m))$.
- To verify the signature s, Bob checks that $h(m) = E_e(s)$.

Forging signed message digests

For Mallory to generate a forged signed message (m', s') he must somehow come up with m' and s' satisfying

$$h(m') = E_e(s') \tag{2}$$

That is, he must find m' and s' that both map to the same string, where m' is mapped by h and s' by E_e.

Two natural approaches for attempting to satisfying (2):

1. Pick m' at random and solve for s'.
2. Pick s' at random and solve for m'.

Approach 1: Solve for s'

Equation:

$$h(m') = E_e(s') \tag{2}$$

To solve for s' given m' requires computing

$$E_e^{-1}(h(m')) = D_d(h(m')) = s'.$$

Alice can compute D_d, which is what enables her to sign messages.

But Mallory presumably cannot compute D_d without knowing d, for if he could, he could also break the underlying cryptosystem.

Approach 2: Solve for m'

Equation:

$$h(m') = E_e(s') \tag{2}$$

To solve for m' given s' requires "inverting" h.

Since h is many-one, a value $y = E_e(s')$ can have many "inverses" or *preimages*.

To successfully forge a signed message, Mallory needs to find only one value m' such that $h(m') = E_e(s')$.

However, the defining property of a cryptographic hash function is that, given y, it should be hard to find any $x \in h^{-1}(y)$.

Hence, Mallory cannot feasibly find m' satisfying 2.

Other attempts

Of course, these are not the only two approaches that Mallory might take.

Perhaps there are ways of generating valid signed messages (m', s') where m' and s' are generated together.

I do not know of such a method, but this doesn't say one doesn't exist.

More advantages of signing message digests

Another advantage of signing message digests rather than signing messages directly: the signatures are shorter.

An RSA signature of m is roughly the same length as m.

An RSA signature of $h(m)$ is a fixed length, regardless of how long m is.

For both reasons of security and efficiency, signed message digests are what is used in practice.

We'll talk more about message digests later on.