Lecture 15

Asymmetric authentication and integrity protection aka Digital signature

Nicola Laurenti November 18, 2020

Lecture 15— Contents

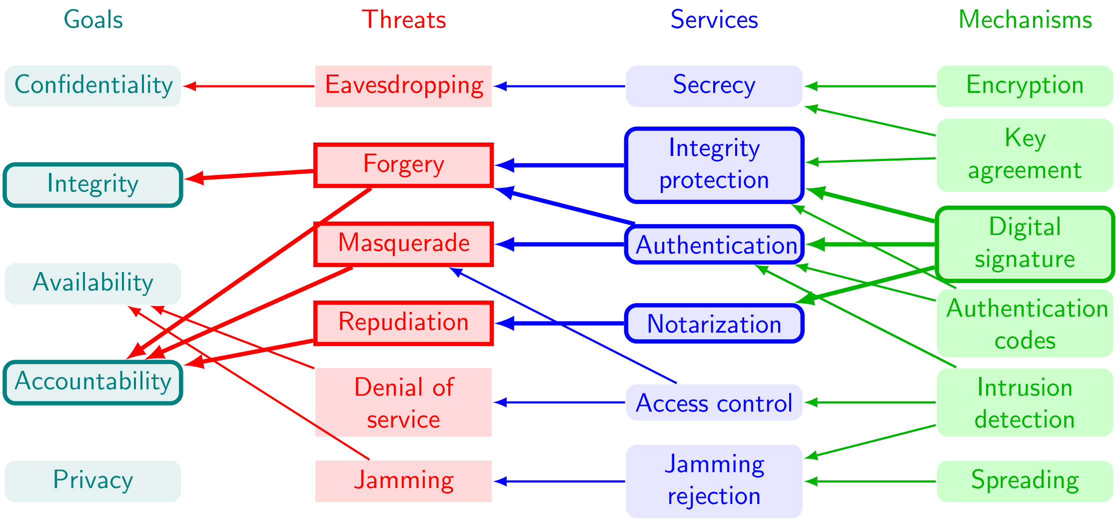
General model for authentication and integrity protection

Examples of digital signature mechanisms

- RSA digital signature
- The Elgamal digital signature

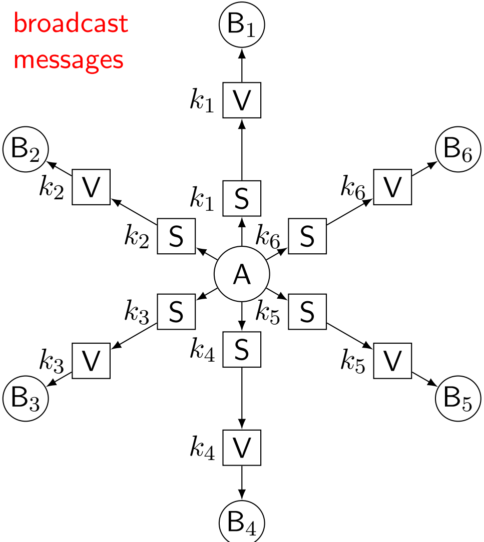
Authenticated encryption

Security goals, threats, services and mechanisms



Motivations for asymmetric authentication + integrity protection

broadcast messages



third party attestation

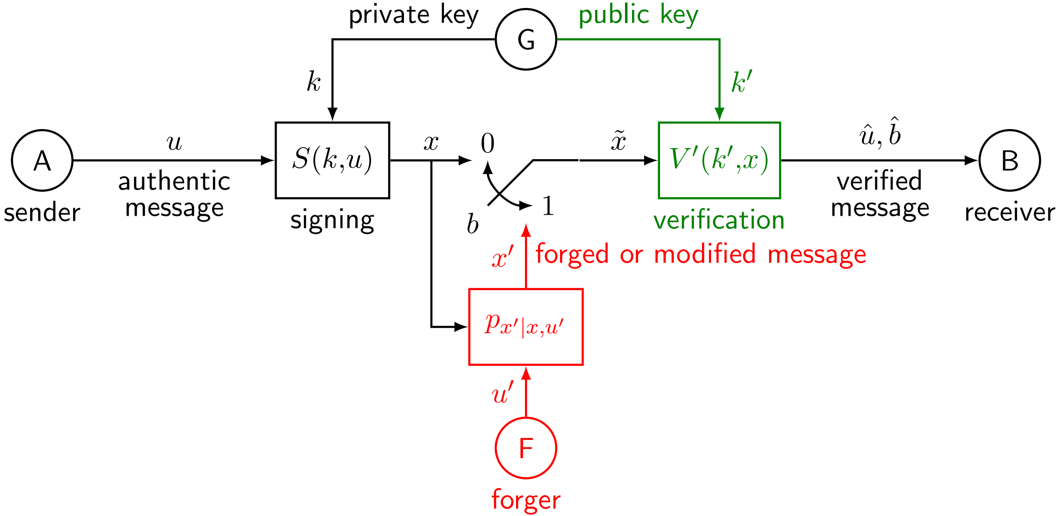
Suppose B is not satisfied with being convinced that u is authentic from A, he also wants to prove it to a third party C.

With symmetric A+IP this cannot be done, because A and B share the same key.

A could repudiate u and accuse B of fabricating the MAC

Is it possible to provide authentication, integrity protection and sender non-repudiation through the same mechanism. That is what a legal signature does.

General model for asymmetric authentication + integrity protection



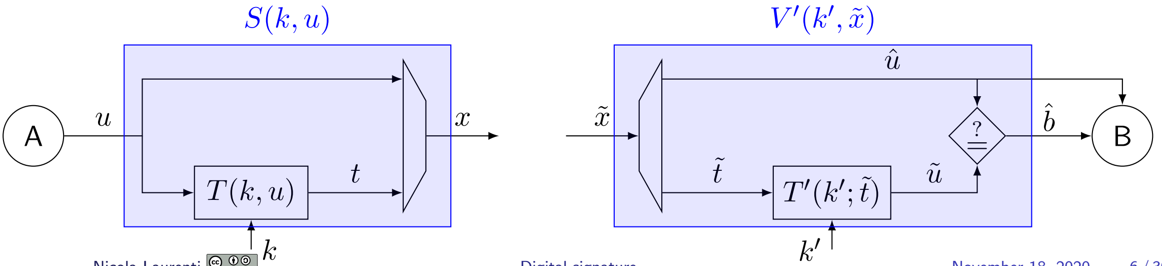
Authentication tags

If the tag appending solution is used for signing

$$x = (u, t) \quad , \qquad t = T(k, u)$$

the verification uses a reparametrization of the inverse tag computation function $T'_{k'} = T_k^{-1}$, recovers the corresponding message from the received tag and checks it against the received message

$$\tilde{x} = (\hat{u}, \tilde{t}) \quad , \qquad \tilde{u} = T'(k', \tilde{t}) \quad , \qquad \hat{b} = \begin{cases} 0 & , \text{ if } \tilde{u} = \hat{u} \\ 1 & , \text{ if } \tilde{u} \neq \hat{u} \end{cases}$$

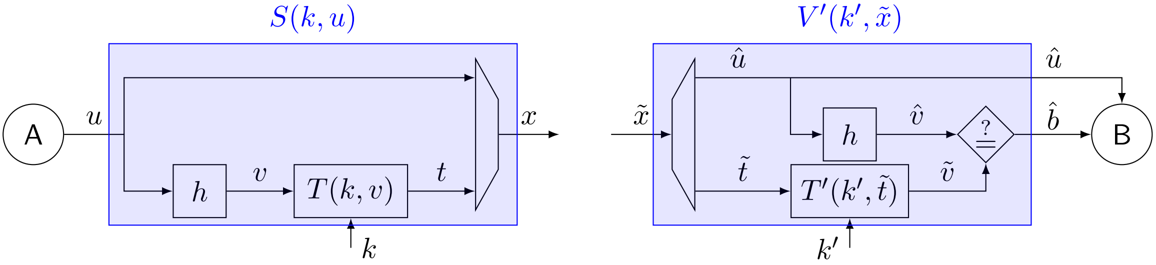


Use of hashing

The above solution has two drawbacks:

1. it only authenticates fixed length messages: in fact, since T_k must be invertible (i.e., injective), it requires $|\mathcal{M}| \leq |\mathcal{T}|$, and $H(u) \leq H(t)$
2. it is vulnerable to existential forgery, where the attacker:
   - generates a random tag $t' \in \mathcal{T}$
   - computes the corresponding message $u' = T'(k', t')$
   - transmits the forged message $x' = (u', t')$

Both problem can be solved by using a cryptographic hash function h



Glossary and notation

private key $k \in \mathcal{K}$ private key space

public key $k' \in \mathcal{K}'$ public key space

message hash $v \in \mathcal{V}$ hash space

received message hash $\tilde{v} \in \mathcal{V}$

verification map $V' : \mathcal{K}' \times \mathcal{X} \mapsto \mathcal{M} \times \{0,1\}$

$$V'_{k'} : \mathcal{X} \mapsto \mathcal{M} \times \{0,1\} \quad V'_{k'}(x) \doteq V'(k', x)$$

hashing function $h : \mathcal{M} \mapsto \mathcal{V}$

tag computation $T : \mathcal{K} \times \mathcal{V} \mapsto \mathcal{T}$

inverse tag map $T' : \mathcal{K} \times \mathcal{T} \mapsto \mathcal{V}$

$$T'_{k'} : \mathcal{T} \mapsto \mathcal{V} \quad T'_{k'}(t) \doteq T'(k', t)$$

Glossary and notation

The asymmetric authentication and integrity protection system is completely specified as:

$$\mathcal{S} = (\mathcal{M}, \mathcal{X}, \mathcal{K}, \mathcal{K}', S, V', p_u, p_k)$$

or, with the appended tag solution, as:

$$\mathcal{S} = (\mathcal{M}, \mathcal{V}, \mathcal{T}, \mathcal{K}, \mathcal{K}', h, T, T', p_u, p_k)$$

In the following examples, we will typically not specify the hashing function h, assuming it is any (well designed) cryptographic hash function

General assumptions

- (correctness) The receiver must be able to recover and accept any authentic message

$$V'_{c'}(S_c(u)) = (u, 0) \quad \forall c \in \mathcal{K}, c' \in \mathcal{K}' : p_{kk'}(c, c') > 0 \quad , \forall u \in \mathcal{M}$$

- (Kerchoff-like assumption) The forger F knows the system $\mathcal{S}$, in particular the maps $S(\cdot,\cdot)$ and $V'(\cdot,\cdot)$, or $h(\cdot)$, $T(\cdot,\cdot)$ and $T'(\cdot,\cdot)$

Where does authenticity come from?

Non forgeability of x is only based on the fact that

1. it is hard to derive k from k' (i.e., f is one-way)
2. it is hard to derive x from (k', u) (i.e., $V'_{k'}$ is one-way)
3. it is hard to derive u from v (i.e., h is one-way)

RSA signature [Rivest-Shamir-Adleman, '77]

Based on NP problems

- integer factorization

  easy given $p, q \in \mathbb{Z}$, compute $n = pq$
  hard given $n \in \mathbb{Z}$, find $p, q \in \mathbb{Z}$ such that $pq = n$

- finite logarithm and finite root

  easy given $n \in \mathbb{Z}$, $x, d \in \mathbb{Z}_n$ compute $y = x^d \bmod n$ (finite exponential)
  hard given $n \in \mathbb{Z}$, $x, y \in \mathbb{Z}_n$ find $d \in \mathbb{Z}_n$ such that $x^d \bmod n = y$
  hard given $n \in \mathbb{Z}$, $d, y \in \mathbb{Z}_n$ find $x \in \mathbb{Z}_n$ such that $x^d \bmod n = y$

The RSA signature scheme

Key generation (ℓ-bit)

A chooses $p, q < 2^{\ell/2}$ primes
computes $n = pq$, $\varphi = (p-1)(q-1)$
chooses $d \in \mathbb{Z}_n$ such that $\gcd(\varphi, d) = 1$
computes $e \in \mathbb{Z}_n$ such that $ed = 1 \pmod{\varphi}$

private key $k = (p, q, d)$, $\mathcal{K} = \mathbb{Z}_{2^\ell}^3$

public key $k' = (n, e)$, $\mathcal{K}' = \mathbb{Z}_{2^\ell}^2$

Signing by A (tag appending, with hash and private key)

$$\mathcal{T} = \mathcal{V} = \mathbb{Z}_n \quad T : \mathcal{K} \times \mathcal{V} \mapsto \mathcal{T}$$

$$t = T(k, v)$$

$$= T(n, d, v) = v^d \bmod n$$

Verification by B (public key)

$$T' : \mathcal{K}' \times \mathcal{T} \mapsto \mathcal{V}$$

$$\tilde{v} = T'(k', \tilde{t})$$

$$= T'(n, e, \tilde{t}) = \tilde{t}^e \bmod n$$

The RSA signature scheme

Theorem (Euler's theorem)

Let $n, \varphi \in \mathbb{Z}$ *as in the key generation and* $u \in \mathbb{Z}_n$. *If* $\gcd(u, n) = 1$, *then* $u^\varphi = 1 \pmod n$

Correctness of RSA signature

We show that if $\tilde{x} = x$ (i.e., $\hat{u} = u, \tilde{t} = t$), then $\tilde{v} = \hat{v}$ and hence $\hat{b} = 0$. Since $\tilde{t} = t$

$$\tilde{v} = \tilde{t}^e = t^e = (v^d)^e = v^{ed} = v^{r\varphi+1} = v$$

with r an arbitrary integer, and thanks to Euler's theorem.
Moreover, since $\hat{u} = u$

$$\hat{v} = h(\hat{u}) = h(u) = v$$

The RSA signature scheme

Computability

choosing p, q primes probabilistic algorithm $O(\ell)$: randomly generate them, then check if primes, else repeat. Probabilistic primality test run in $O(\ell)$ (e.g., Fermat test), the fastest deterministic primality test (Lenstra-Pomerance variant of the AKS test) has complexity $O(\ell^6)$ (still prohibitive)

computing n, φ is $O(\ell)$

choosing d probabilistic algorithm $O(\ell)$: randomly generate d, then check if coprime with φ, else repeat. Coprimality can be tested with Euclidean algorithm that is $O(\ell)$

computing e can be done with Euclidean algorithm

signing and verification finite exp $O(\ell^2)$ (typically, $e \ll n$, so verification is fast)

The RSA signature scheme

Security

$v = T'_{k'}(t)$ is one-way finding t from v and e is hard (finite root)

$k' = f(k)$ is one-way finding d from e, without knowing φ is hard

finding φ from n is hard (no easier than finding p, q)

finding p, q from n is hard (integer factorization)

$t = T(\cdot, v)$ is one-way finding d from (v, t) is hard (finite logarithm)

The RSA signature scheme

Necessity of hashing

Besides existential forgery, hashing also prevents the following chosen message attack

- F knows one pair $x = (t, u)$ with $t = T(k, u) = u^d \bmod n$ and aims to forge $x' = (u', t')$, for some target forged message u', with $t' = T(k, u') = (u')^d \bmod n$.
- he computes $u_1 = u'u \bmod n$ and lures A into signing u_1, obtaining

$$t_1 = (u_1)^d = (u'u)^d = (u')^d u^d = t't \pmod{n}$$

- he derives $t' = t_1 t^{-1} \bmod n$

With hashing the attacker can compute

$$h(u')h(u) \bmod n = v'v \bmod n = v_1 = h(u_1)$$

but can not derive u_1 (preimage reistance)

The Elgamal signature scheme [Elgamal, ‘85] aka digital signature algorithm / standard (DSA, DSS)

Based on NP problem

finite logarithm

easy given $n \in \mathbb{Z}$, $x, d \in \mathbb{Z}_n$ compute $y = x^d \bmod n$ (finite exponential)

hard given $n \in \mathbb{Z}$, $x, y \in \mathbb{Z}_n$ find $d \in \mathbb{Z}_n$ such that $x^d \bmod n = y$

Key generation

Let p be a prime, and α a primitive element in $(\mathbb{Z}_p, \cdot)$, i.e. such that $\forall \beta \in \mathbb{G}, \exists n : \alpha^n = \beta$.
Assume p and α are publicly known

private and public key space $\mathcal{K} = \mathcal{K}' = \mathbb{Z}_p$

A generates $k \sim \mathcal{U}(\mathcal{K})$, then computes $k' = f(k) = \alpha^k \bmod p$

The Elgamal signature scheme

Signing by A (tag appending, with hash and private key, probabilistic)

$\mathcal{V} = \mathbb{Z}_p \quad , \quad \mathcal{T} = \mathbb{Z}_p \times \mathbb{Z}_{p-1}$

A generates $r \sim \mathcal{U}(\mathbb{Z}_{p-1})$ with $\gcd(r, p-1) = 1$

$$t = T_k(v, r) = (t_1, t_2) \quad , \quad \begin{cases} t_1 = \alpha^r \bmod p \\ t_2 = (v - kt_1)r^{-1} \bmod (p-1) \end{cases}$$

Verification by B (public key, non standard tag appending)

B need not know r

$$\begin{aligned} \hat{b} &= V'(k', x) = V'(k', u, t_1, t_2) \\ &= \begin{cases} 0 & , \text{ if } k'^{t_1} t_1^{t_2} = \alpha^{h(u)} \pmod p \\ 1 & , \text{ otherwise} \end{cases} \end{aligned}$$

The Elgamal signature scheme

Theorem (Fermat's little theorem)

Let p *be a prime and* $\alpha \in \mathbb{Z}$ *such that* $\gcd(\alpha, p-1) = 1$*.*
Then, $\alpha^{(p-1)} = 1 \pmod p$

Correctness

We prove that if $\tilde{x} = x$ (i.e., $\hat{u} = u, \tilde{t}_1 = t_1, \tilde{t}_2 = t_2$), then $k'^{\tilde{t}_1} \tilde{t}_1^{\tilde{t}_2} = \alpha^{h(\hat{u})} \pmod p$ and hence $\hat{b} = 0$.

$$k'^{\tilde{t}_1} \tilde{t}_1^{\tilde{t}_2} = k'^{t_1} t_1^{t_2} = \alpha^{kt_1} (\alpha^r)^{t_2} = \alpha^{kt_1 + rt_2}$$

Since $rt_2 = v - kt_1 \pmod{p-1}$, we have $kt_1 + rt_2 = v + \ell(p-1)$ for some integer ℓ and

$$\alpha^{kt_1+rt_2} \bmod p = \alpha^v \alpha^{\ell(p-1)} \bmod p = \alpha^v \left(\alpha^{(p-1)}\right)^\ell \bmod p$$

Then, by Fermat's little theorem, and by the fact that $u = \hat{u}$

$$\alpha^{kt_1+rt_2} \bmod p = \alpha^v 1^\ell \bmod p = \alpha^v = \alpha^{h(u)} = \alpha^{h(\hat{u})}$$

The Elgamal signature scheme

Security

verification is one-way given v and k', but not k nor r, it is hard to find t

$k' = f(k)$ is one-way finding k from k' is hard (finite log problem)

$T(\cdot, v)$ is one-way given t and v, but not r it is hard to find k from the equation $kt_1 + rt_2 = v \pmod{p-1}$

Importance of r

secret if attacker learns r, he can solve the equation $kt_1 + rt_2 = v \pmod{p-1}$ for k

varied if A uses the same r to sign both u and u', then $t_1 = t'_1$ and

$$v - v' = k(t_1 - t'_1) + r(t_2 - t'_2) = r(t_2 - t'_2) \pmod{p-1}$$

attacker can learn r from v, v', t_2, t'_2, then back to the above

The Elgamal signature scheme

Importance of hashing

For Elgamal signatures there is a different existential forgery attack than in the standard case.
Let $q \in \mathbb{Z}_{p-1}$, then consider $t' = (t'_1, t'_2)$ with

$$\begin{cases} t'_1 = k'\alpha^q \bmod p \\ t'_2 = -t'_1 \bmod (p-1) \end{cases}$$

At verification,

$$(k')^{t'_1}(t'_1)^{t'_2} = (k')^{t'_1}(k')^{t'_2}\alpha^{qt'_2} = (k')^{t'_1}(k')^{-t'_1}\alpha^{qt'_2} = \alpha^{qt'_2} \pmod p$$

Then, without hashing, t' would be a correct signature for message $u' = qt'_2 \bmod (p-1)$

Variants of the Elgamal signature scheme

All based on the finite log problem

Digital signature algorithm (DSA)

Instead of $p-1$, one of its prime factors q is used, and $t_1 = (\alpha^r \bmod p) \bmod q$

Schnorr signature

Stronger security proof, under the assumption that hashing is ideal.

Nyberg-Rueppel signature

u can be recovered from t (message recovery property), redundancy instead of hashing against existential forgery

Elliptic curve DSA

An important variant of the Elgamal signature is the elliptic curve version

Uses an elliptic curve group $(\mathcal{E}, \circ)$ on a field $\mathbb{F} = \mathbb{Z}_p$ with p prime, with cardinality $|\mathcal{E}| = q$ and a primitive point $P \in \mathcal{E}$, and mixes operations on both $\mathbb{Z}_q$ and $\mathcal{E}$.

The use of elliptic curve cryptography increases the security wrt DSA for the same key length.

Hence ECDSA is very appropriate when short keys are needed (especially the public key that must be distributed)

Example

For the same level of security SL $= 80$ bits ECDSA uses a 160-bit key, DSA and RSA use 1024-bit keys

Beware the importance of r

In the Sony PS3 software signature cracking (2010), the private key employed by Sony to sign the PS3 software was exposed because the same $r = 4$ was reused

Elliptic curve DSA

Key generation

private key space $\mathcal{K} = \{1, \ldots, q-1\}$, public key space $\mathcal{K}' = \mathcal{E}$

A randomly generates $k \sim \mathcal{U}(\mathcal{K})$, computes $k' = P \overset{k}{\circ}$

Denote by $c_1(Q) \in \mathbb{F}$ the first coordinate of a point $Q \in \mathbb{F}^2$

Signing (private, probabilistic)

$$\mathcal{T} = \mathcal{K}^2 \; , \; \mathcal{V} = \mathbb{Z}_q$$

A generates $r \sim \mathcal{U}(\mathcal{K})$

hashes message $v = h(u)$

computes $t = (t_1, t_2) \in \mathcal{T}$

$$\begin{cases} t_1 = c_1(P \overset{r}{\circ}) \bmod q \\ t_2 = (v + k t_1) r^{-1} \bmod q \end{cases}$$

Verification (public key)

B computes

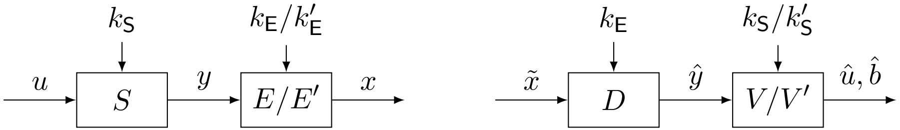
$$\begin{cases} m = t_2^{-1} h(u) \bmod q \\ n = t_2^{-1} t_1 \bmod q \\ Q = (P \overset{m}{\circ}) \circ (k' \overset{n}{\circ}) \end{cases}$$

$$\hat{b} = \begin{cases} 0 & , \text{ if } c_1(Q) = t_1 \pmod{q} \\ 1 & , \text{ otherwise} \end{cases}$$

Sign-then-encrypt

What if we want to both keep our message secret and guarantee is authenticity and integrity (aka build a secure channel) ?

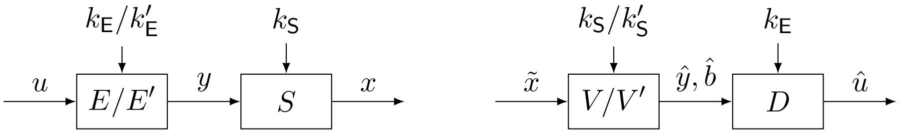
One possible solution is to first sign the information message, then encrypt the signed message



- the two mechanisms (E, D) and (S, V) can be separately designed, each with its target security requirements
- needs two distinct key pairs
- can use symmetric or asymmetric mechanisms both for signature and encryption
- was used in the Transport Layer Security (TLS) Record protocol but turned out vulnerable to padding oracle attacks

Encrypt-then-sign

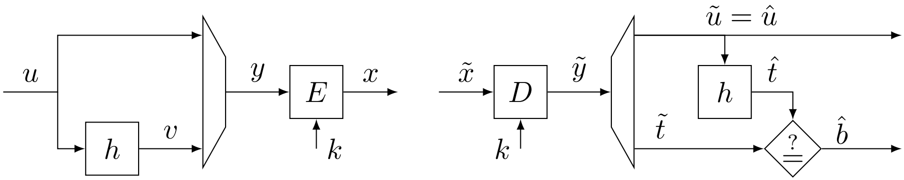
Another possible solution is to first encrypt the information message, then sign the encrypted message



- the two mechanisms (E, D) and (S, V) can be separately designed, each with its target security requirements
- needs two distinct key pairs
- if a message is not accepted $(\hat{b} = 1)$, do not decrypt it: save workload and avoid padding oracle attacks
- can use symmetric or asymmetric mechanisms both for signature and encryption
- used in the Internet Protocol Security (IPSec) Encapsulating Security Payload (ESP) protocol and in the TLS Encrypt-then-MAC extension

Hash-then-encrypt

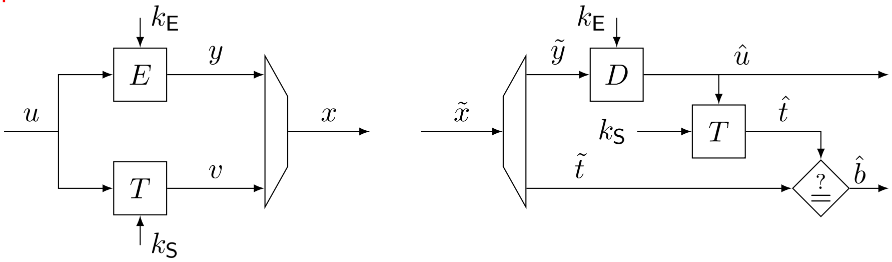
Another possible solution is to first hash the information message, then encrypt the concatenation of message and its hash



- encryption acts also as signature
- needs only one key pair
- cannot use asymmetric mechanisms (public encryption $\Rightarrow$ public signing, public verification $\Rightarrow$ public decryption)
- used in the ill-famed IEEE 802.11 Wired Equivalent Privacy (WEP) protocol (epic fail: the hash was a linear CRC code and the encryption was the RC4 additive stream cipher)

Sign-and-encrypt

Another possible solution is to compute the authentication tag on the plaintext, then append it to the ciphertext



- the two mechanisms (E, D) and T can be separately designed, each with its target security requirements
- needs two distinct key pairs
- can use symmetric or asymmetric mechanisms
- in asymmetric signature, cryptographic hashing is needed to avoid revealing $u = T'(k', t)$ (beside preventing existential forgery)

Summary

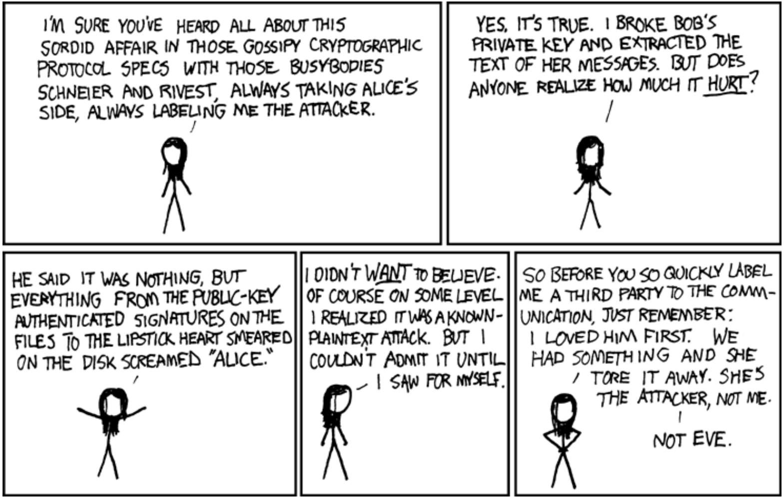
In this lecture we have:

- formulated a general model for asymmetric message authentication and integrity protection, aka digital signature
- described two examples of digital signature mechanisms: RSA and DSA
- discussed ways to combine encryption and authentication

Assignment

- class notes
- textbook, §8.1 – §8.4, §8.7

End of lecture



Alice and Bob, reproduced from xkcd URL: xkcd.com/177