

CSE 207B: Applied Cryptography

Nadia Heninger

UCSD

Fall 2025 Lecture 12

Announcements

1. HW 5 is due!
2. HW 6 is due in 1 week.

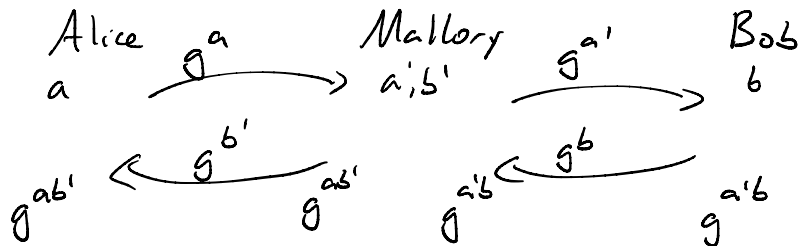
Last time:

- Attacks on RSA

This time:

- Digital signatures

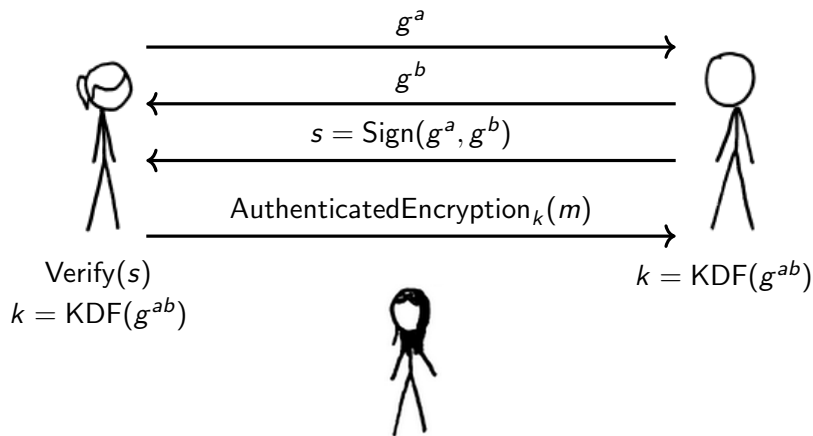
Man-in-the-middle attacks against Diffie-Hellman



An active man-in-the-middle attacker can modify Diffie-Hellman key exchange messages in transit, and Alice and Bob will never know.

Public Key Idea #3: Digital Signatures

Solving the authentication problem.



* Still a toy protocol; not secure.

Digital Signatures

- Key generation: Generate key pair (pk, sk)
- Sign: $\text{sign}_{sk}(m) = \sigma$
- Verify:

$$\text{verify}_{pk}(m, \sigma) = \begin{cases} \text{accept} & \text{if valid} \\ \text{reject} & \text{otherwise} \end{cases}$$

Correctness: $\Pr[\text{verify}_{pk}(m, \text{sign}_{sk}(m)) = \text{accept}] = 1.$

Digital Signatures

- Key generation: Generate key pair (pk, sk)
- Sign: $\text{sign}_{sk}(m) = \sigma$
- Verify:

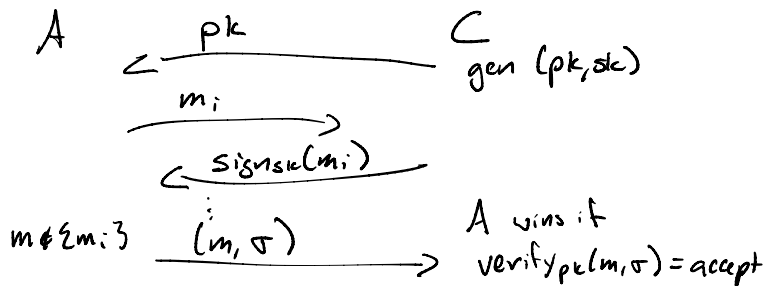
$$\text{verify}_{pk}(m, \sigma) = \begin{cases} \text{accept} & \text{if valid} \\ \text{reject} & \text{otherwise} \end{cases}$$

Correctness: $\Pr[\text{verify}_{pk}(m, \text{sign}_{sk}(m)) = \text{accept}] = 1.$

Differences between handwritten signatures and cryptographic signatures:

- Only person with secret key can generate signature.
- Anyone with public key can verify signature.
- Cryptographic signature is a function of the signed message.

Signature security forgery game



Definition

A signature scheme is existentially unforgeable under a chosen-message attack if $\Pr[A \text{ wins}]$ is negligible.

Differences between signatures and MACs

- Signatures are publicly verifiable
- MACs are non-binding: either Alice or Bob could have generated
- Signatures are usually non-repudiable
- Signatures might allow an adversary to create a new valid pair (m, σ') from (m, σ) , which isn't allowed for a MAC
- Signer may not know what m is

Digital signature applications

- Authenticated key exchange
- Software signing
- Digital certificates and public key infrastructure
- Email spam prevention (DKIM)
- Cryptocurrencies

Textbook RSA Signatures

[Rivest Shamir Adleman 1977]

Public Key

$N = pq$ modulus

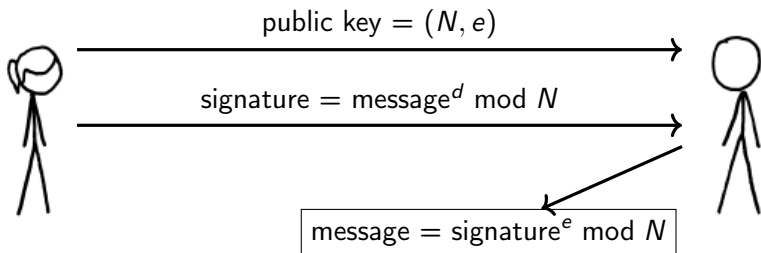
e verifying exponent

Private Key

p, q primes

d signing exponent

$(d = e^{-1} \bmod (p-1)(q-1))$



Textbook RSA signatures are insecure

Forgery from public key:

1. Choose arbitrary value σ .
2. Output $(m = \sigma^e \bmod N, \sigma)$.

Textbook RSA signatures are insecure

Arbitrary forgery for any m :

1. Pick r . Compute $m' = r^e m \bmod N$
2. Query signature for m' : $\sigma' = (r^e m)^d \bmod N = r m^d$
3. Output $(m, \sigma' r^{-1} \bmod N)$.

Signature blinding: Allows signer to sign without seeing message.

“Hash-and-sign” paradigm

- Allows signing of arbitrary length messages.
- Protects against signature forgery

Use a secure signature scheme ($\text{sign}, \text{verify}$) and a collision-resistant hash function H .

- Key generation: Generate key pair (pk, sk)
- Sign': $\sigma = \text{sign}_{sk}(H(m))$
- Verify':

$$\text{verify}'_{pk}(m, \sigma) = \begin{cases} \text{accept} & \text{if } \text{verify}_{pk}(H(m), \sigma) \\ \text{reject} & \text{otherwise} \end{cases}$$

Theorem

If the signature scheme is existentially unforgeable under adaptive chosen-message attack and the hash function is collision resistant, then this construction is existentially unforgeable under an adaptive chosen-message attack.

PKCS #1 v.1.5 padding for RSA signatures

SHA256 has output length 256, but we would like to use it with 2048-bit RSA keys.

PKCS#1 v. 1.5 padding is commonly used.

$$m = 00\ 01\ [FF\ FF\ \dots FF]\ 00\ [DI]\ [H(m)]$$

DI specifies the hash function that was used, and $H(m)$ is the hash of the message to be signed.

Bleichenbacher RSA signature forgery attack

PKCS#1 v. 1.5 padding:

$$m = 00\ 01\ [FF\ FF\ \dots FF]\ 00\ [DI]\ [H(m)]$$

Bleichenbacher noticed that implementations that:

1. Use small public exponent e
2. Do not check length of $FF\ FF\ FF\dots$ padding

Would be vulnerable to signature forgery.

Attack: Generate value r such that

$$r^e = 00\ 01\ FF\ 00\ DI\ H(m)\ ||\ \text{garbage over } \mathbb{Z}.$$

Padding length is wrong, but verifier doesn't notice.

More secure approach: "full-domain hashing":

$H : \{0, 1\}^* \rightarrow \mathbb{Z}/N\mathbb{Z}$ for e.g. 2048-bit N . Needs new hash constructions.

FIPS PUB 186-3

**FEDERAL INFORMATION PROCESSING STANDARDS
PUBLICATION**

Digital Signature Standard (DSS)

CATEGORY: COMPUTER SECURITY

SUBCATEGORY: CRYPTOGRAPHY

DSA (Digital Signature Algorithm)

Public Key

p prime

q prime, divides $(p - 1)$

g generator of subgroup of
order $q \bmod p$

$$y = g^x \bmod p$$

Private Key

x private key

Verify

$$u_1 = H(m)s^{-1} \bmod q$$

$$u_2 = rs^{-1} \bmod q$$

$$r \stackrel{?}{=} g^{u_1} y^{u_2} \bmod p \bmod q$$

Sign

Generate random k .

$$r = g^k \bmod p \bmod q$$

$$s = k^{-1}(H(m) + xr) \bmod q$$

DSA security

Theorem (Pointcheval, Vaudenay 96)

Breaking DSA is equivalent to computing discrete logs in the random oracle model.

Prime-field DSA is not used much anymore. RSA historically more common for SSL/TLS.

Elliptic curve DSA is becoming much more common: used for SSH, growing in popularity for TLS.

DSA Nonce Vulnerability

Public Key

p prime

q prime, divides $(p - 1)$

g generator of subgroup of
order $q \bmod p$

$$y = g^x \bmod p$$

Private Key

x private key

Sign

Generate random k .

$$r = g^k \bmod p \bmod q$$

$$s = k^{-1}(H(m) + xr) \bmod q$$

The nonce k must remain secret, or else the secret key x is revealed.

$$x = (sk - H(m))r^{-1} \bmod q$$

DSA Repeated Nonce Vulnerability

Public Key

p prime

q prime, divides $(p - 1)$

g generator of subgroup of
order $q \bmod p$

$y = g^x \bmod p$

Private Key

x private key

Sign

Generate random k .

$r = g^k \bmod p \bmod q$

$s = k^{-1}(H(m) + xr) \bmod q$

If k is ever reused to sign distinct message hashes $H(m_1)$, $H(m_2)$,
it is easy to compute:

$$k = (H(m_1) - H(m_2))(s_1 - s_2)^{-1} \bmod q$$

Then the secret key can be computed as before:

$$x = (s_1 k - H(m_1))r_1^{-1} \bmod q$$

DSA Repeated Nonce Vulnerabilities in the Wild

- 1% of SSH DSA host keys compromised by poor RNGs in 2012.
- 2013 Android RNG issue caused Bitcoin thefts due to ECDSA repeated nonce vulnerability.
- Sony PS3 code signing key computed in 2011 due to fixed nonce used to sign ECDSA

Countermeasure: Use “deterministic” nonce generation:

$$k = H(m||x)$$

or “hedged” nonce generation:

$$k = H(m||x||\text{random})$$

(As standardized, H is HMAC_0)