Introduction to public-key cryptography

Part 2: applications of public-key cryptography

Jean-Sébastien Coron

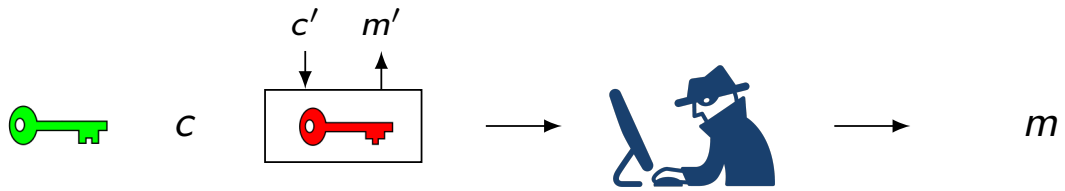
University of Luxembourg

Outline

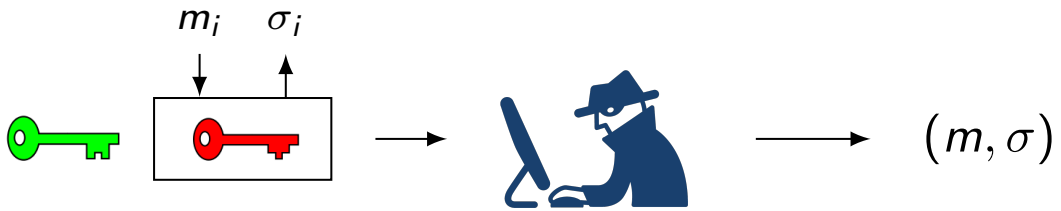
- Part 1: basic constructions (previous lecture)
  - History
  - Classical cryptography: block-ciphers, hash functions
  - Public-key cryptography: RSA encryption and RSA signatures, DH key exchange
- Part 2: applications of public-key cryptography (this lecture)
  - Security models
  - How to encrypt and sign securely with RSA. OAEP and PSS.
  - Public-key infrastructure. Certificates, HTTPS protocol.

From primitives to secure applications

- Textbook RSA encryption is deterministic and malleable: guessing and chosen-ciphertext attacks can reveal information.



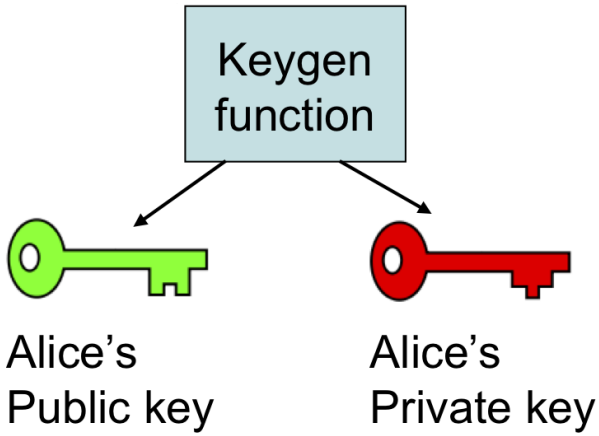
  - OAEP: a provably secure encoding for RSA.
- Textbook RSA signatures permit forgeries



  - Use a secure hash-and-sign encoding such as PSS.
- Diffie–Hellman is vulnerable to a man-in-the-middle
  - Authenticate the exchange to bind it to the intended peer.
- This lecture: security models, secure schemes, certificates and TLS.

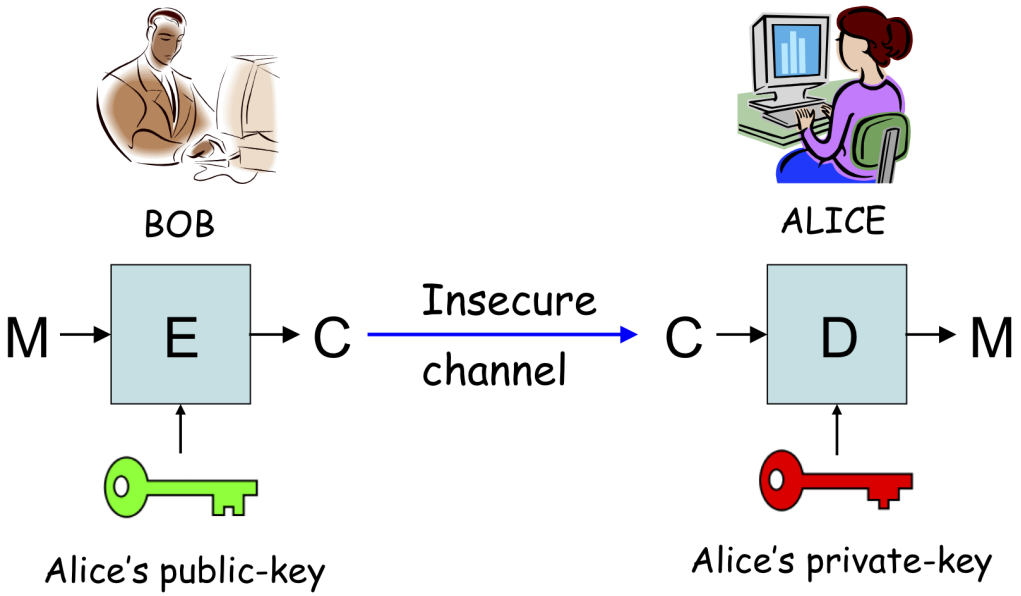
Public-key cryptography

- Invented by Diffie and Hellman in 1976. Revolutionized the field.
- Each user now has two keys
  - A public key
  - A private key
  - Should be hard to compute the private key from the public key.
- Enables:
  - Asymmetric encryption
  - Digital signatures
  - Key exchange, identification, and many other protocols.



Public-key encryption

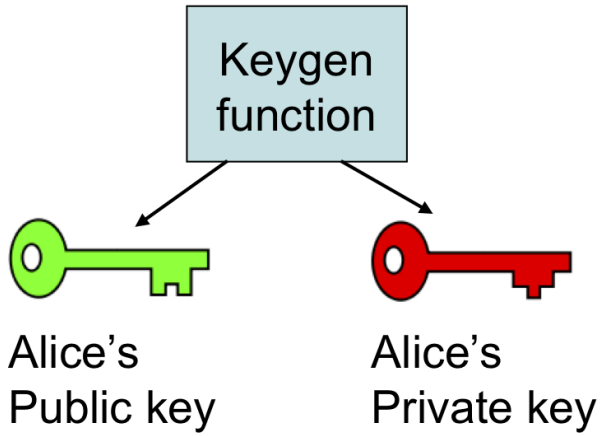
- Public-key encryption (or asymmetric encryption)
  - Enables encryption without a pre-shared secret key
  - The recipient's public key still needs to be authenticated



- Key generation:
  - Generate two large distinct primes p and q of same bit-size $k/2$, where k is a parameter.
  - Compute $n = p \cdot q$ and $\phi = (p-1)(q-1)$.
  - Select a random integer e such that $\gcd(e, \phi) = 1$
  - Compute the unique integer d such that

$$e \cdot d \equiv 1 \pmod{\phi}$$

    using the extended Euclidean algorithm.
  - The public key is (n, e).
  - The private key is d.



Textbook RSA encryption (insecure)

- Encryption with public-key (n, e)
  - Given a message $m \in [0, n-1]$ and the recipent's public-key (n, e), compute the ciphertext:

$$c = m^e \bmod n$$

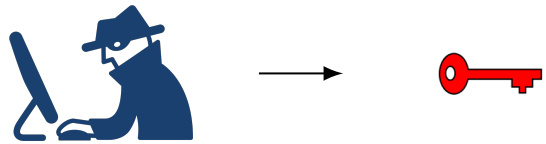
- Decryption with private-key d
  - Given a ciphertext c, to recover m, compute:

$$m = c^d \bmod n$$

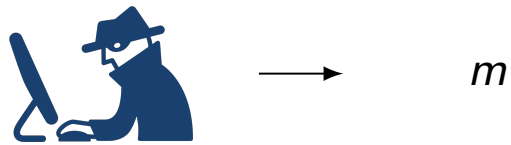
- Security is based on the hardness of factorization
  - Given $n = p \cdot q$, no known efficient algorithm to recover the primes p and q.
  - Public modulus n must be large enough: use at least 2048 bits.

Security models

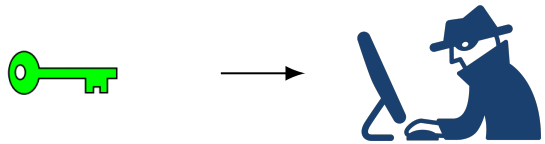
- To be rigorous when speaking about security, one must specify: the attacker's goal and the attacker's power.
- The attacker's goal
  - Does he need to recover the private key ?



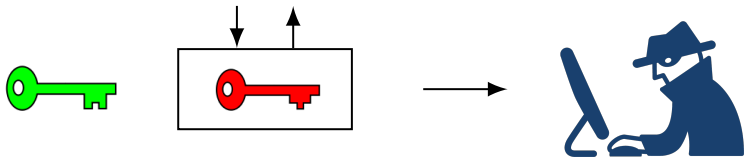
  - or only decrypt a particular ciphertext (or less) ?



- The attacker's power
  - Does he get only the user's public-key ?

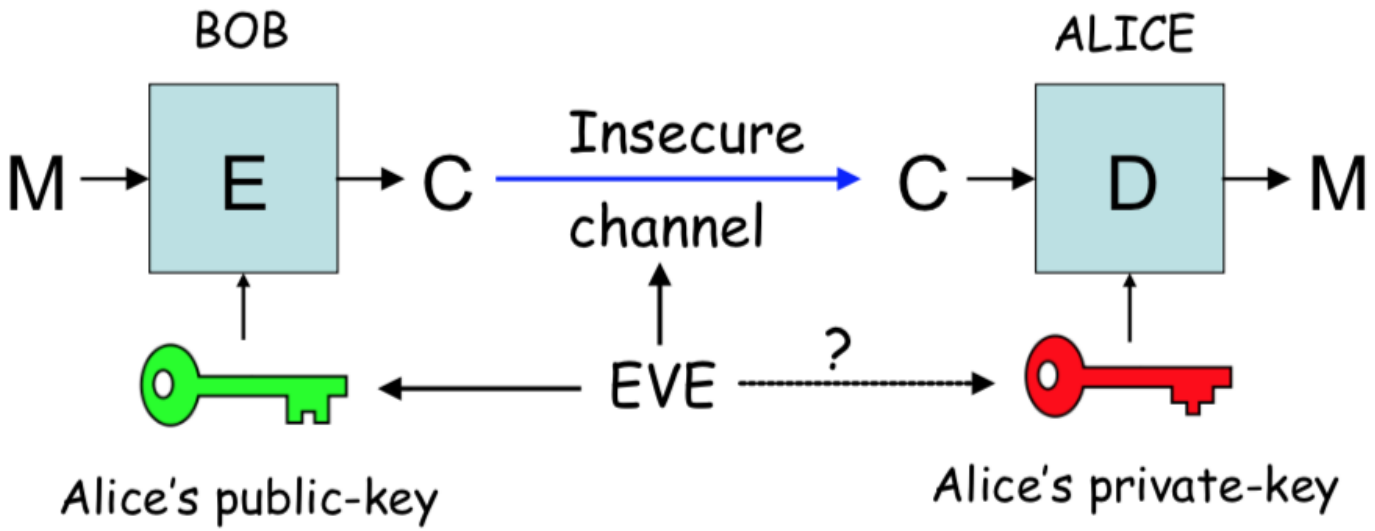


  - or more ?



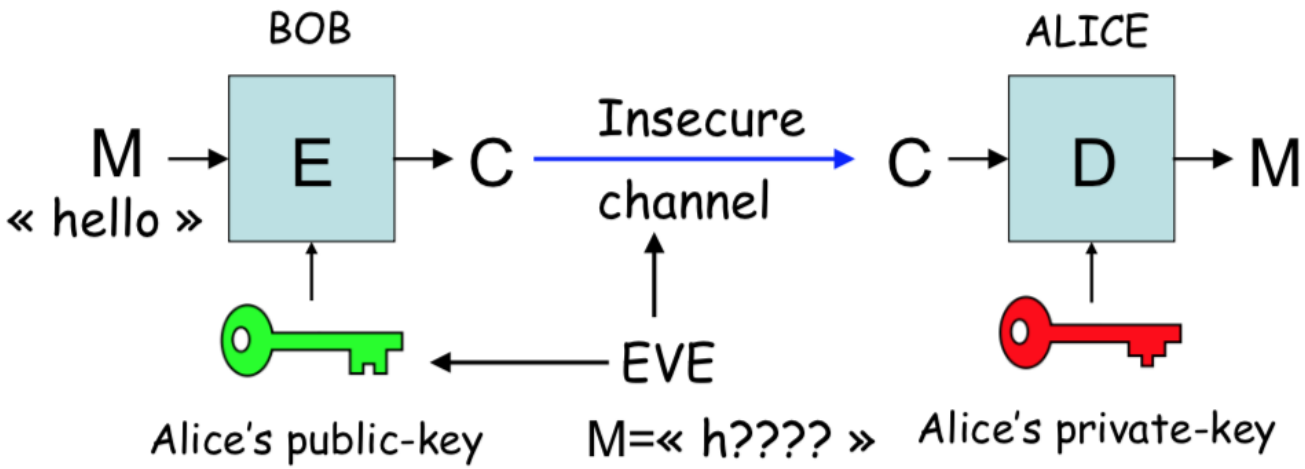
Attacker's goal

- One may think that the adversary's goal is always to recover the private key.
  - complete break
  - may be too ambitious in practice



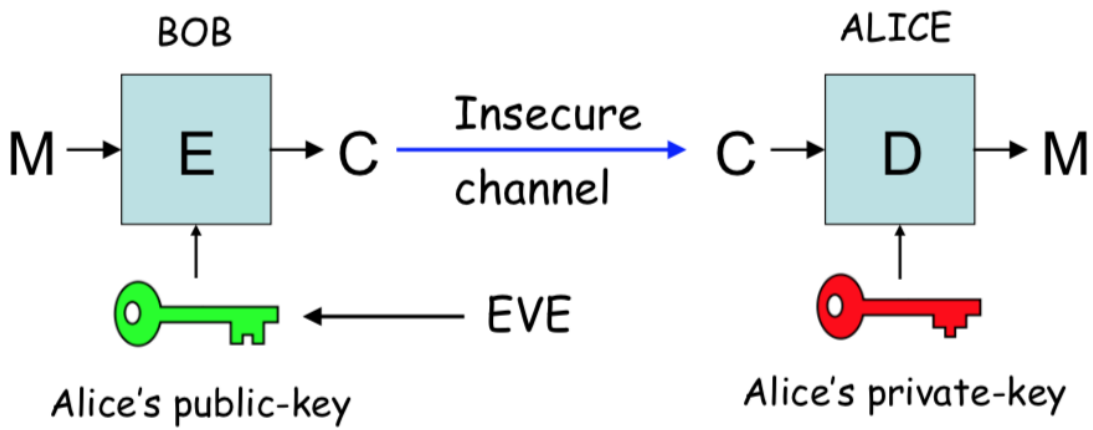
Attacker's goal

- More modest goal: being able to decrypt one ciphertext.
  - or recover some information about a plaintext (for example, the first character)



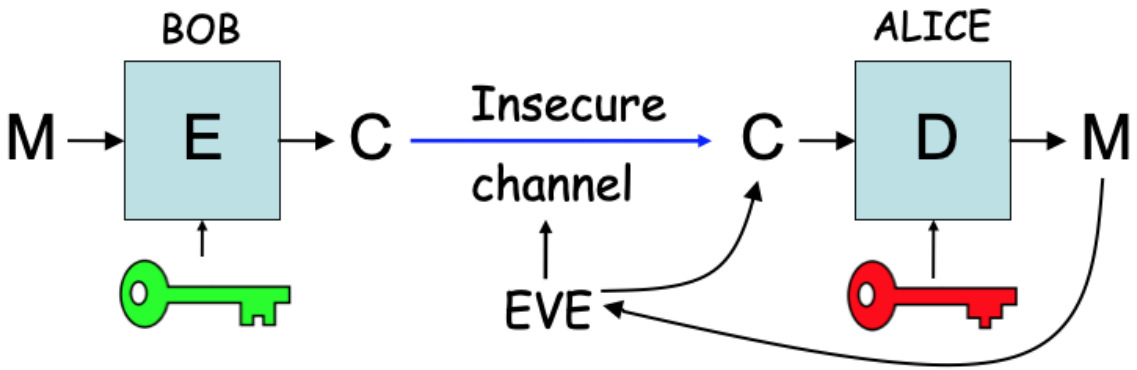
Attack model

- Specify the power of the attacker
- Public-key only attack: the attacker gets only the public-key
  - Called chosen plaintext attack (CPA) because the adversary can encrypt any plaintext of his choice.
  - Weakest adversary



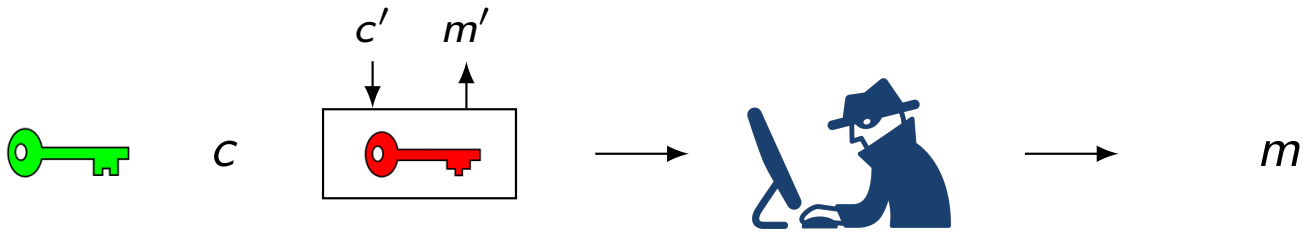
Chosen ciphertext attack (CCA)

- Most powerful attack
- The attacker can obtain decryption of ciphertexts of his choice
- May be realistic in practice
  - attacker gets access to a decryption machine
  - encryption algorithm used in a more complex protocol in which users can obtain decryption of chosen ciphertexts.



Chosen ciphertext attack against textbook RSA

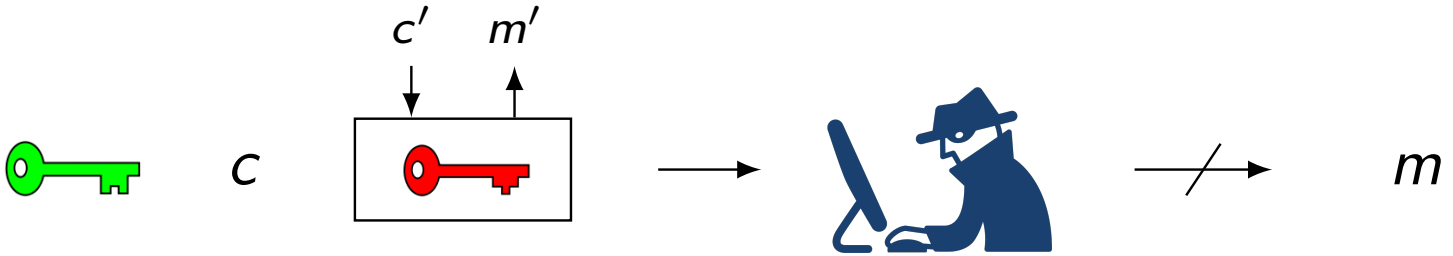
- Chosen-ciphertext attack:
  - Given ciphertext c to be decrypted
  - Generate a random r
  - Ask for the decryption of the random looking ciphertext $c' = c \cdot r^e \pmod{n}$
  - One gets $m' = (c')^d = c^d \cdot (r^e)^d = c^d \cdot r = m \cdot r \pmod{n}$
  - This enables to compute $m = m'/r \pmod{n}$



- Conclusion: do not use textbook RSA encryption !

IND-CCA2 security

- Indistinguishability under adaptive chosen ciphertext attack (IND-CCA2)
  - Formalized in 1991 by Rackoff et Simon
  - A ciphertext should hide the plaintext, even with access to a decryption oracle (except on the challenge ciphertext).
  - A standard security notion for public-key encryption.



The IND-CCA2 security game



Adversary

Challenger

Public key pk

Decryption queries c and answers $\mathrm{Dec}(sk, c)$

m_0, m_1 with $|m_0| = |m_1|$

Choose $b \leftarrow \{0, 1\}$
$c^* = \mathrm{Enc}(pk, m_b)$

Challenge c^*

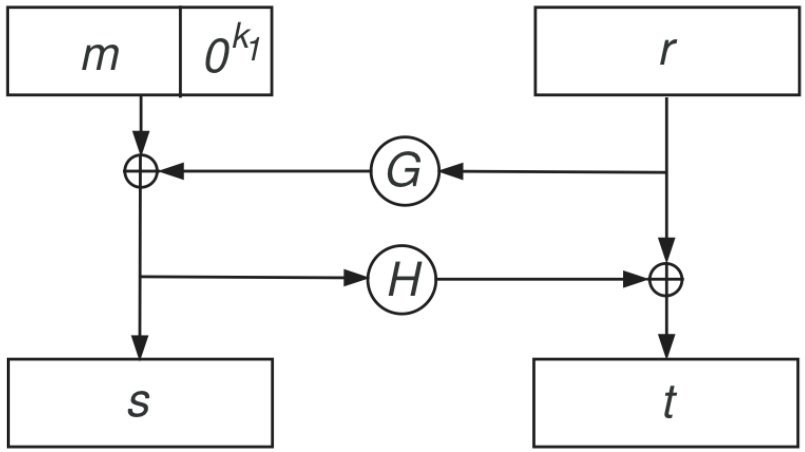
Decryption queries $c \neq c^*$ and answers $\mathrm{Dec}(sk, c)$

Output guess b'

- Advantage: $\left|\Pr[b' = b] - \frac{1}{2}\right|$.
- IND-CCA2 security: every efficient adversary has negligible advantage.
  - An efficient attacker cannot tell which message was encrypted with probability meaningfully above $1/2$.

OAEP

- OAEP (Bellare and Rogaway, E'94)
  - IND-CCA2, assuming that RSA is hard to invert, in the random oracle model.
  - PKCS #1 v2.1

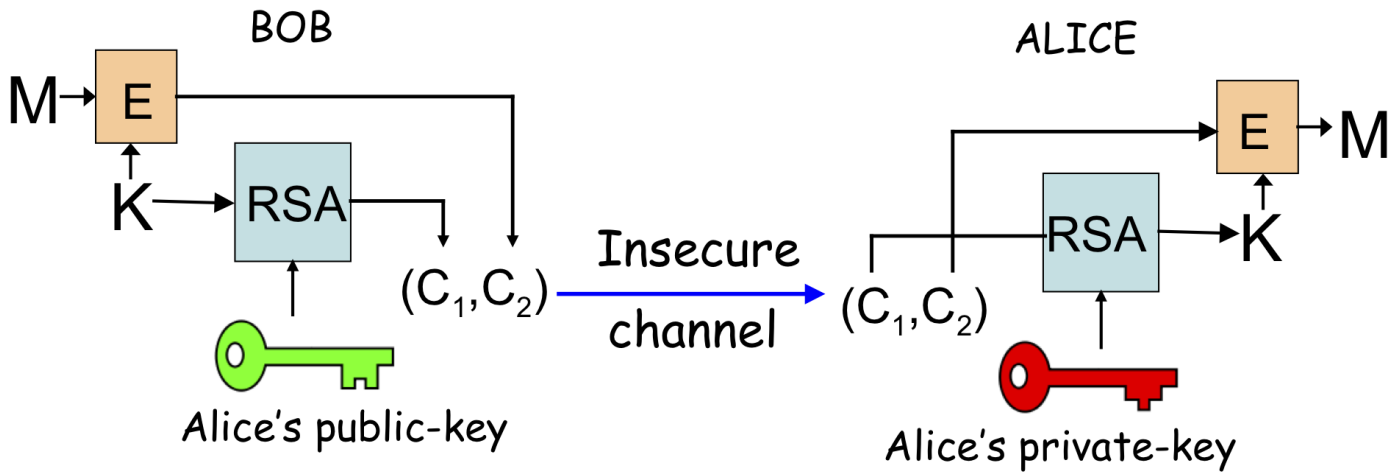


$$c = (s||t)^e \bmod N$$

- Message length limitation
  - If N has k bits and $|r| = k_0$, then $|m| \leq k - k_0 - k_1$ bits.
  - For larger m, use hybrid encryption.

Hybrid encryption with RSA-OAEP

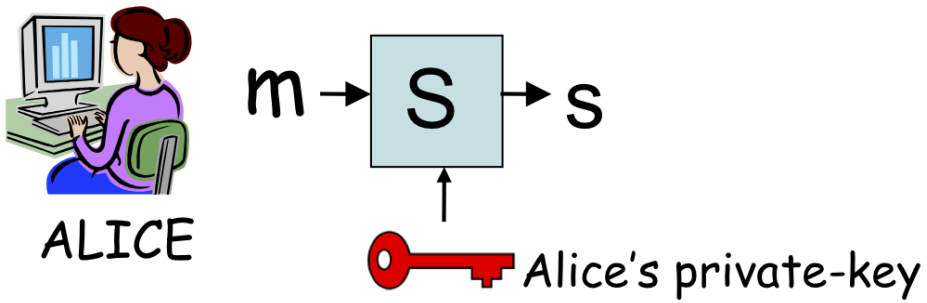
- To encrypt a long message m, Bob generates a fresh symmetric key K.
  - Encrypt K using RSA-OAEP and Alice's public key: C_1.
  - Encrypt m using authenticated encryption (e.g. AES-GCM) with K: C_2.



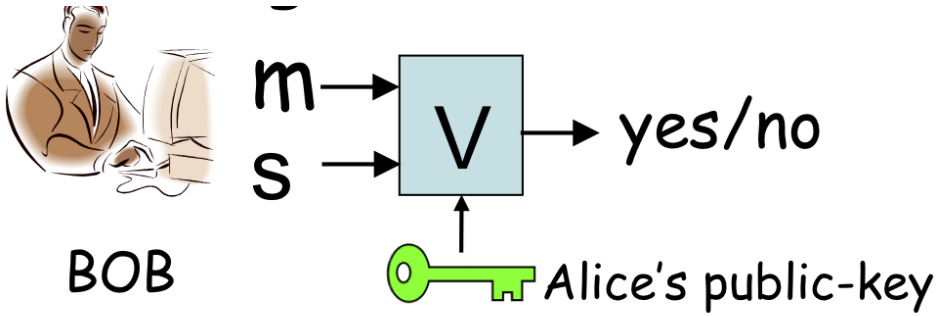
- Send (C_1, C_2). Alice decrypts C_1 to recover K, then verifies and decrypts C_2.

Digital signatures

- A digital signature σ is a bit string that depends on the message m and the user's public-key pk
  - Only Alice can sign a message m using her private-key sk



  - Anybody can verify Alice's signature of the message m given her public-key pk



The RSA signature scheme

- Key generation :
  - Public modulus: $N = p \cdot q$ where p and q are large primes.
  - Public exponent : e
  - Private exponent: d, such that $d \cdot e = 1 \pmod{\phi(N)}$
- To sign a message m, the signer computes :
  - $s = m^d \bmod N$
  - Only the signer can sign the message.
- To verify the signature, one checks that:
  - $m = s^e \bmod N$
  - Anybody can verify the signature

Attacks against textbook RSA signatures

- Given $\sigma_1 = (m_1)^d \bmod N$ and $\sigma_2 = (m_2)^d \bmod N$:
  - one can compute the signature of $m_1 \cdot m_2$ without knowing d:

$$\sigma = (m_1 \cdot m_2)^d = (m_1)^d \cdot (m_2)^d = \sigma_1 \cdot \sigma_2 \pmod{N}$$

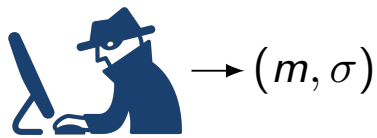
- One cannot use plain RSA signature
  - Hash-and-sign paradigm: the message is first hashed

$$m \longrightarrow H(m) \longrightarrow 1001 \ldots 0101 \| H(m)$$
$$\downarrow$$
$$\sigma = \left(1001 \ldots 0101 \| H(m)\right)^d \pmod{N}$$

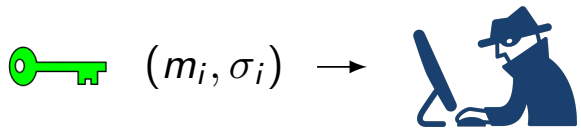
Attack scenario for signature schemes

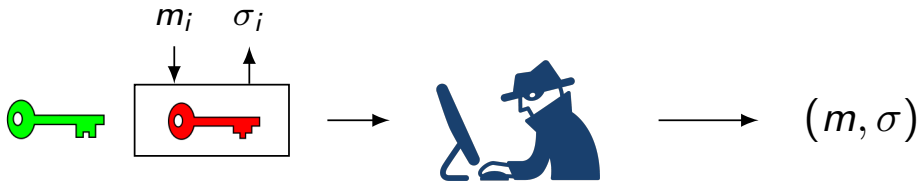
- We must specify the adversary's goal and the adversary's power.
- Adversary's goal
  - Controlled forgery: the adversary can produce the signature of any message
  - Existential forgery: the adversary can produce the signature of a (possibly meaningless) message





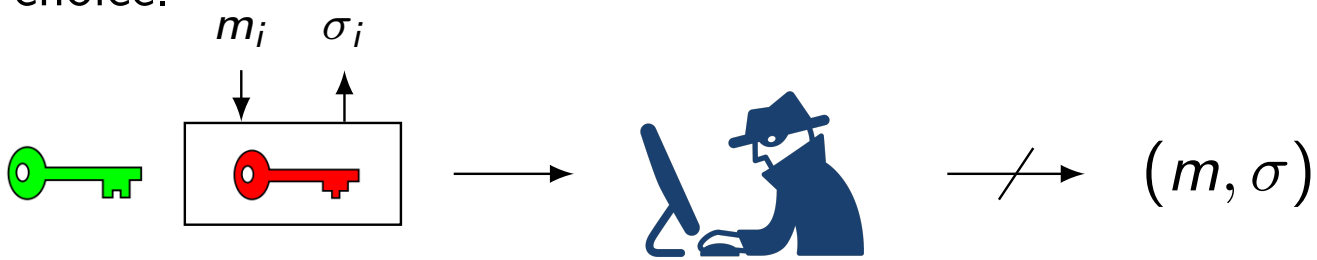
- Adversary's power
  - Known message attack: the adversary obtains a set of pairs message/signature
  - Chosen message attack: the adversary can obtain the signature of any message of his choice, adaptively.





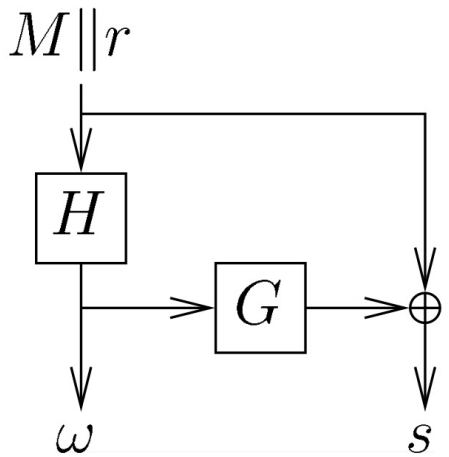
Security notion for signature scheme

- Combines weakest goal with strongest adversary
- Existential unforgeability under an adaptive chosen message attack
  - Defined by Goldwasser, Micali and Rivest in 1988
  - It must be infeasible for an attacker to forge the signature of a message, even if he can obtain signatures of messages of his choice.



The PSS signature scheme

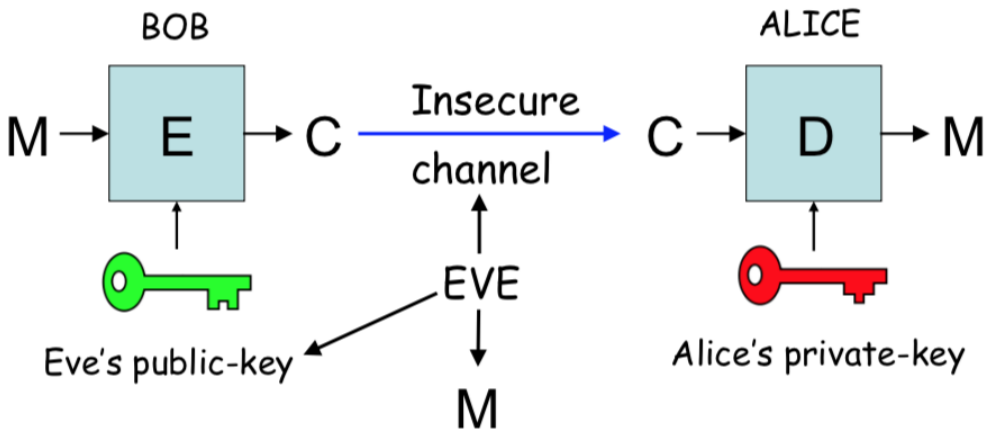
- PSS (Bellare and Rogaway, Eurocrypt'96)
  - IEEE P1363a and PKCS#1 v2.1.
  - 2 variants: PSS and PSS-R (message recovery)
  - Provably secure against chosen-message attacks, in the random oracle model.
  - PSS-R: $\mu(M, r) = \omega \| s$, $\sigma = \mu(M, r)^d \bmod N$



- TLS 1.3 uses RSA-PSS for RSA handshake signatures (CertificateVerify).

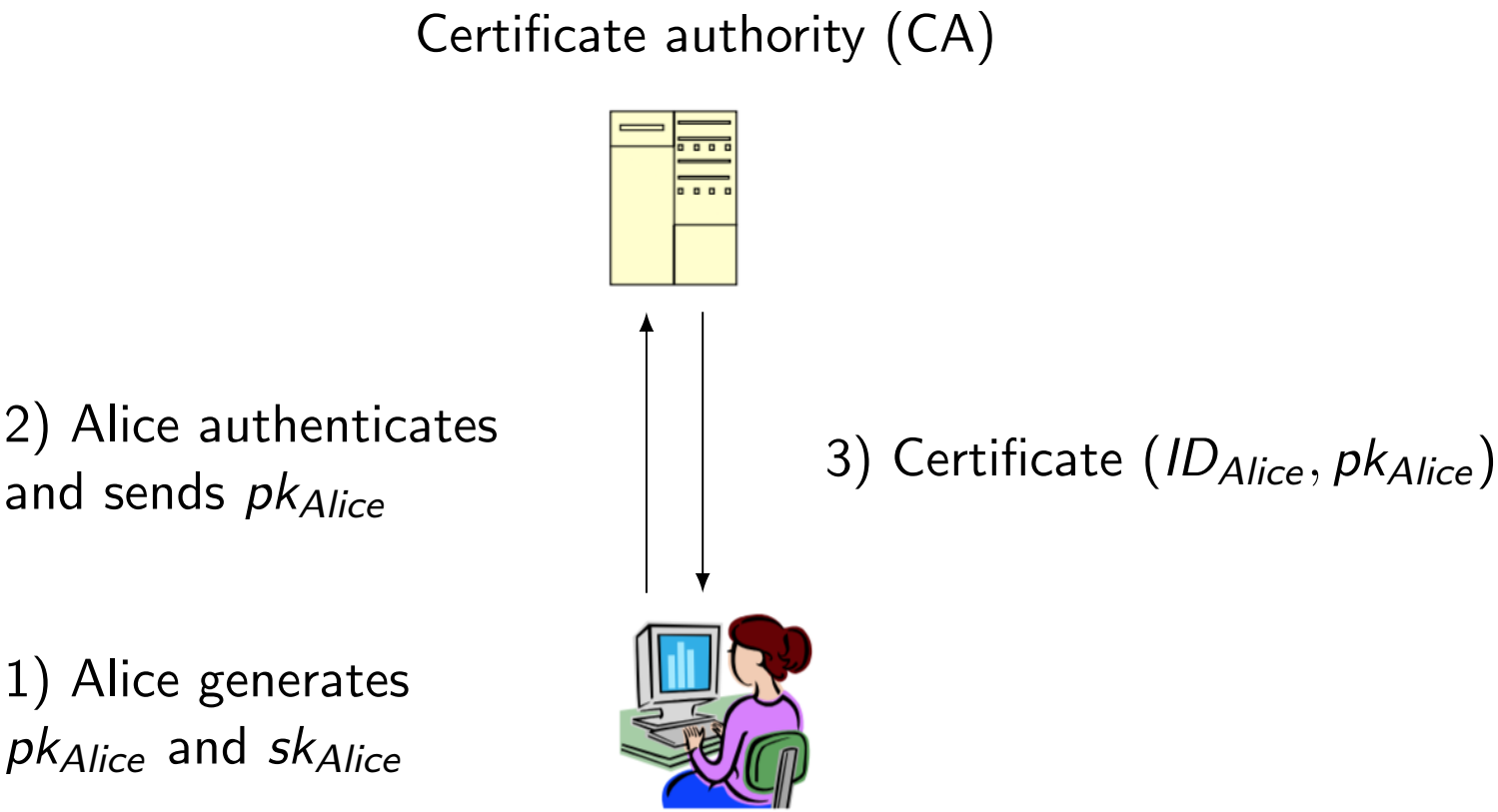
Public-key infrastructure

- Encryption and signatures rely on an authentic public key
  - Bob needs to know that the public key really belongs to Alice.
  - Otherwise, an attacker can substitute a key and impersonate Alice.
  - A certificate binds an identity to a public key through a CA signature.
  - Bob must validate the certificate using a trusted CA key.



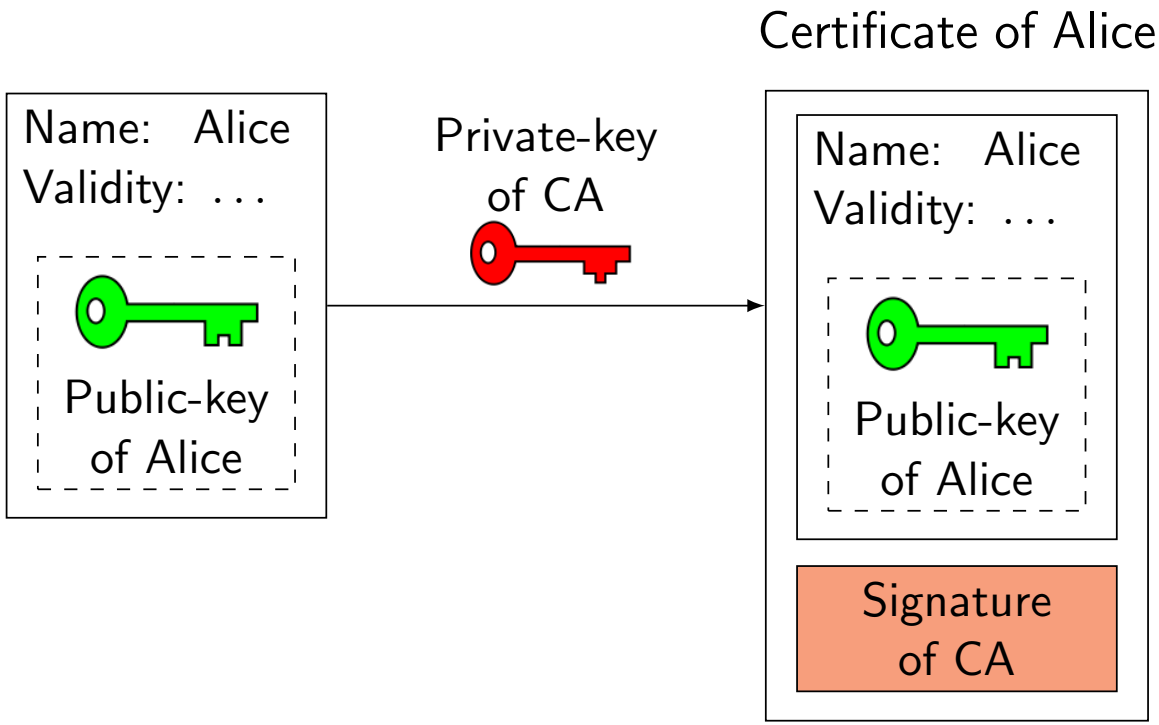
Public-key Infrastructure (PKI)

- Certificate authorities (CAs) bind public keys to identities.
  - A CA signs a certificate containing an identity and public key
  - Validation relies on a trusted CA and helps prevent key substitution

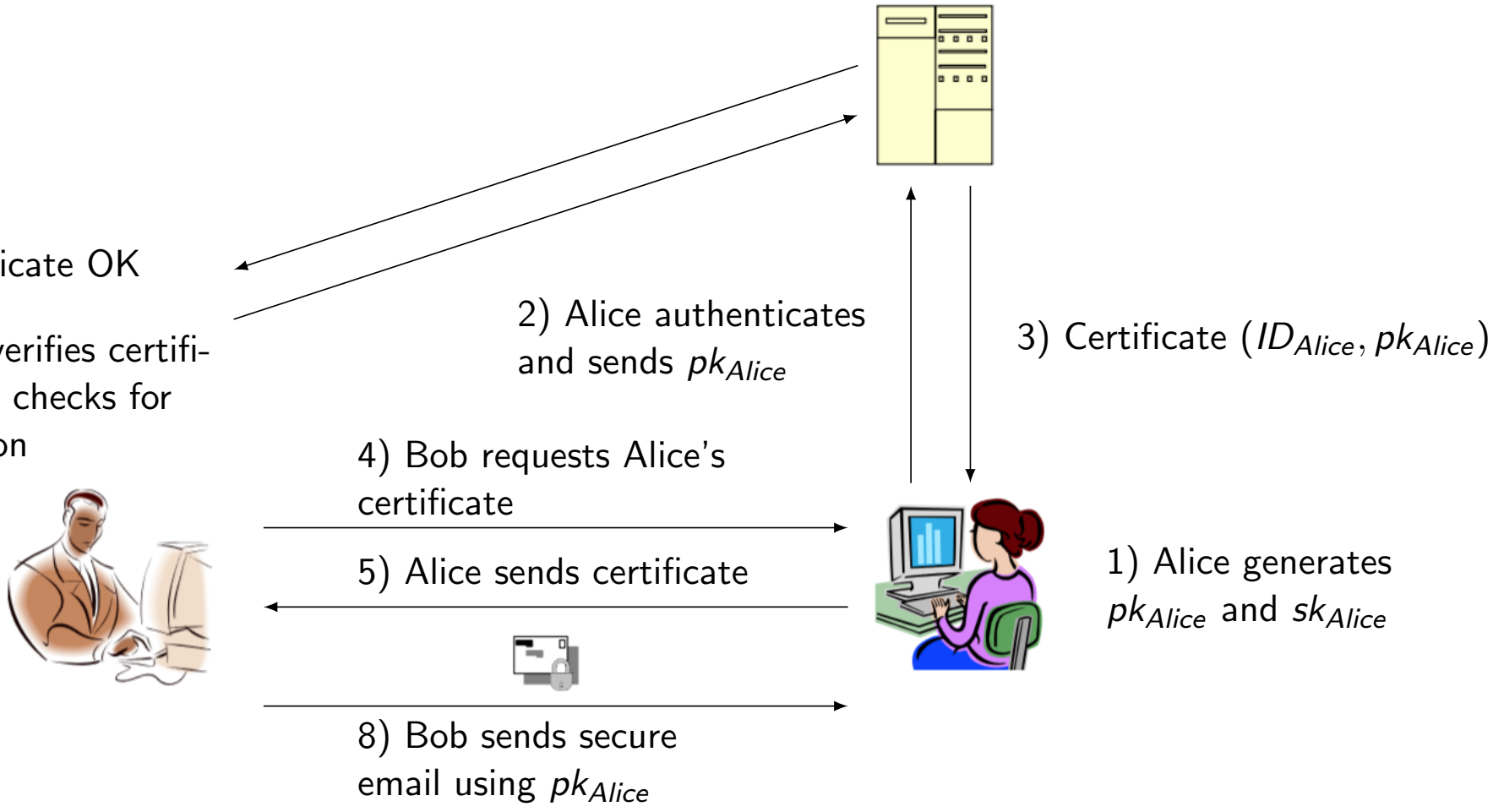


Public-key certificate

- The signature of the certificate authority (CA) binds together a public-key with an identity in the certificate.
  - Proves ownership of a public-key
  - Bob can be sure that the public-key belongs to Alice by checking the signature using the CA public-key.
  - The CA is trusted by all participants.
  - Most common certificate standard: X.509

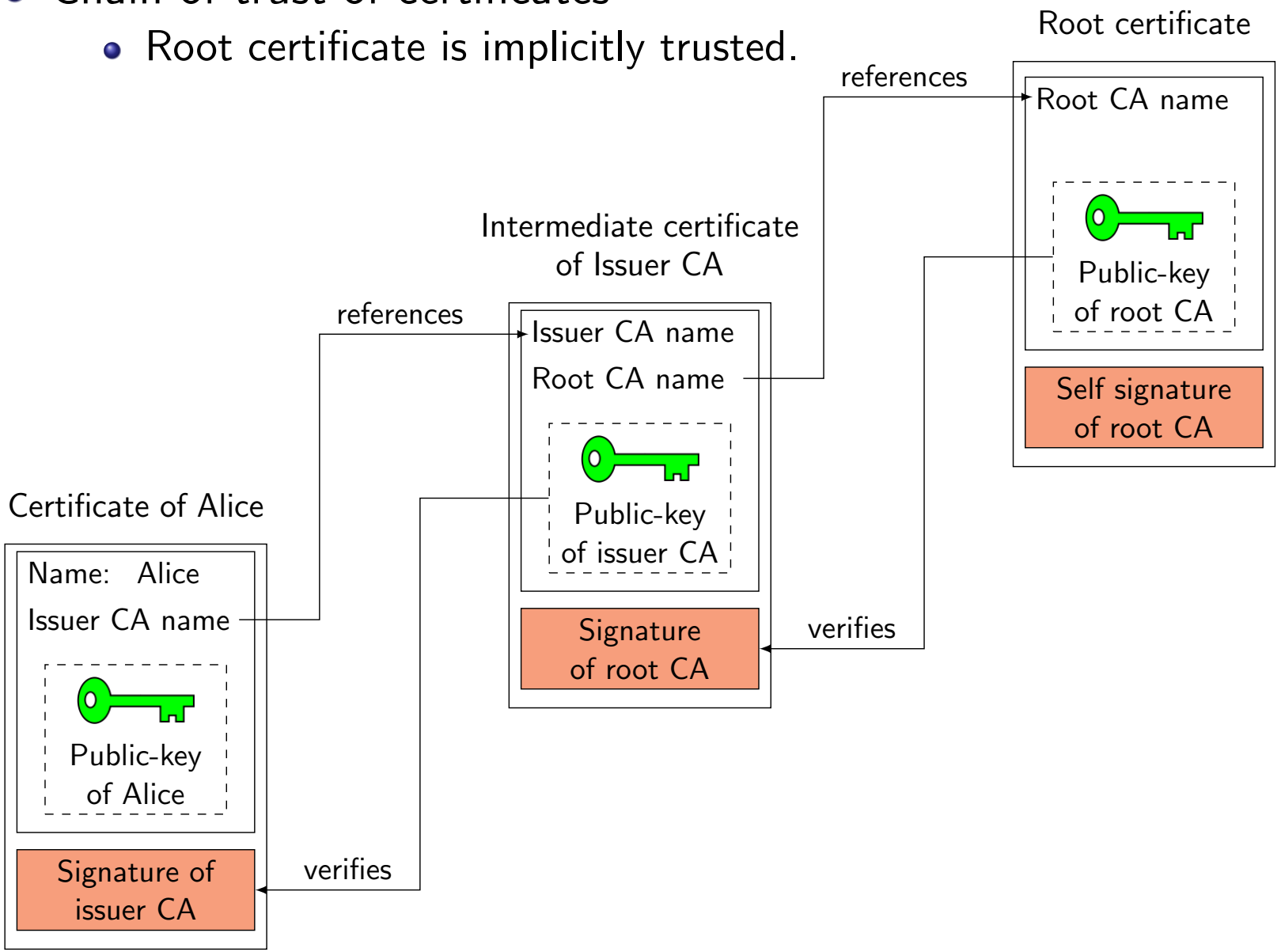


PKI encryption



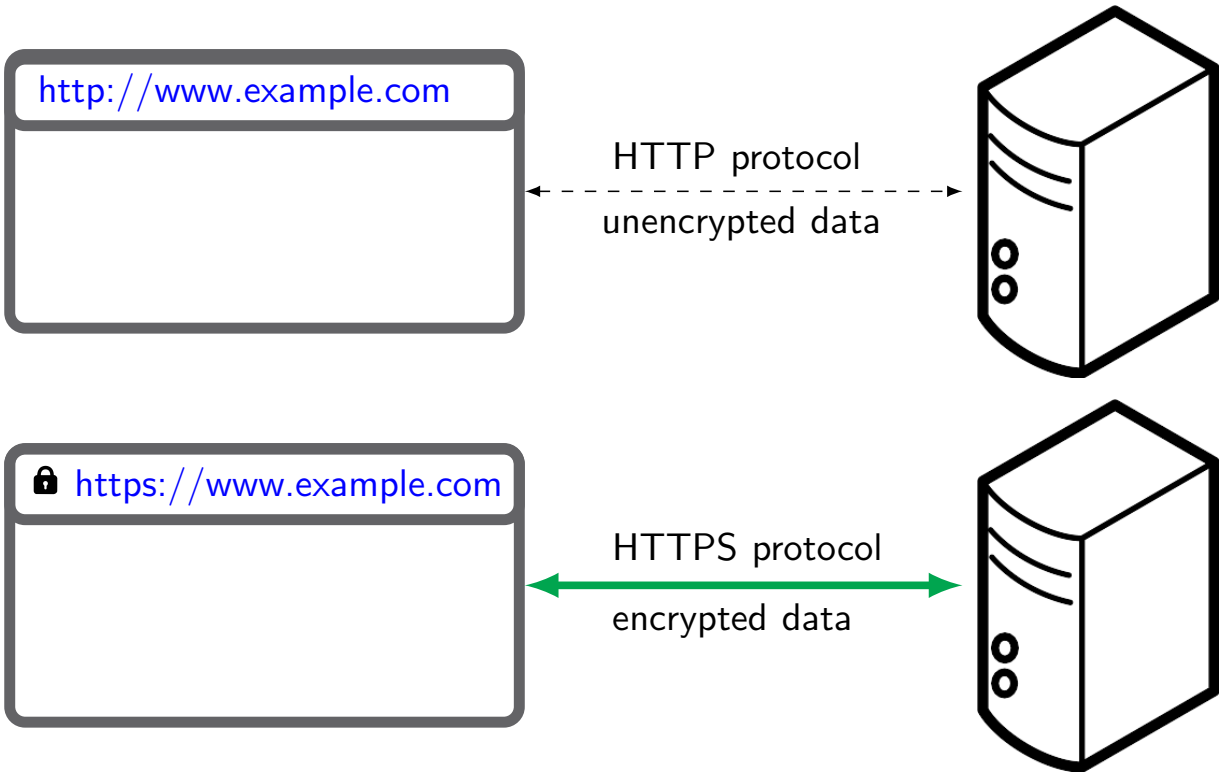
Hierarchy of certificates

- Chain of trust of certificates
  - Root certificate is implicitly trusted.



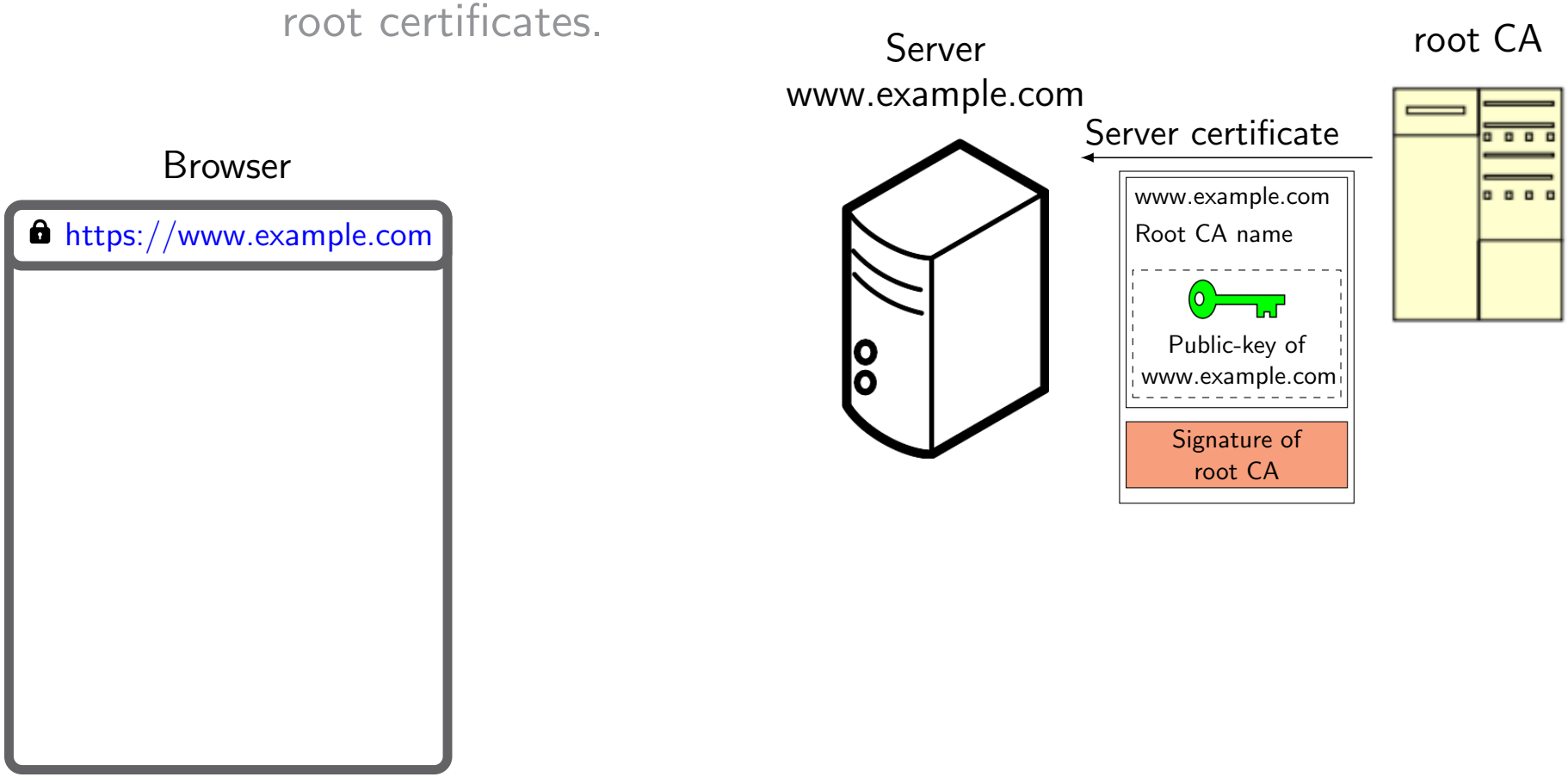
The HTTPS protocol

- Hypertext Transfer Protocol Secure (HTTPS) is a protocol for securely browsing the web.
  - Communication is encrypted using Transport Layer Security (TLS); formerly, Secure Sockets Layer (SSL).
  - Validates the authenticity of an HTTPS web server, and ensures confidentiality and integrity of communications.



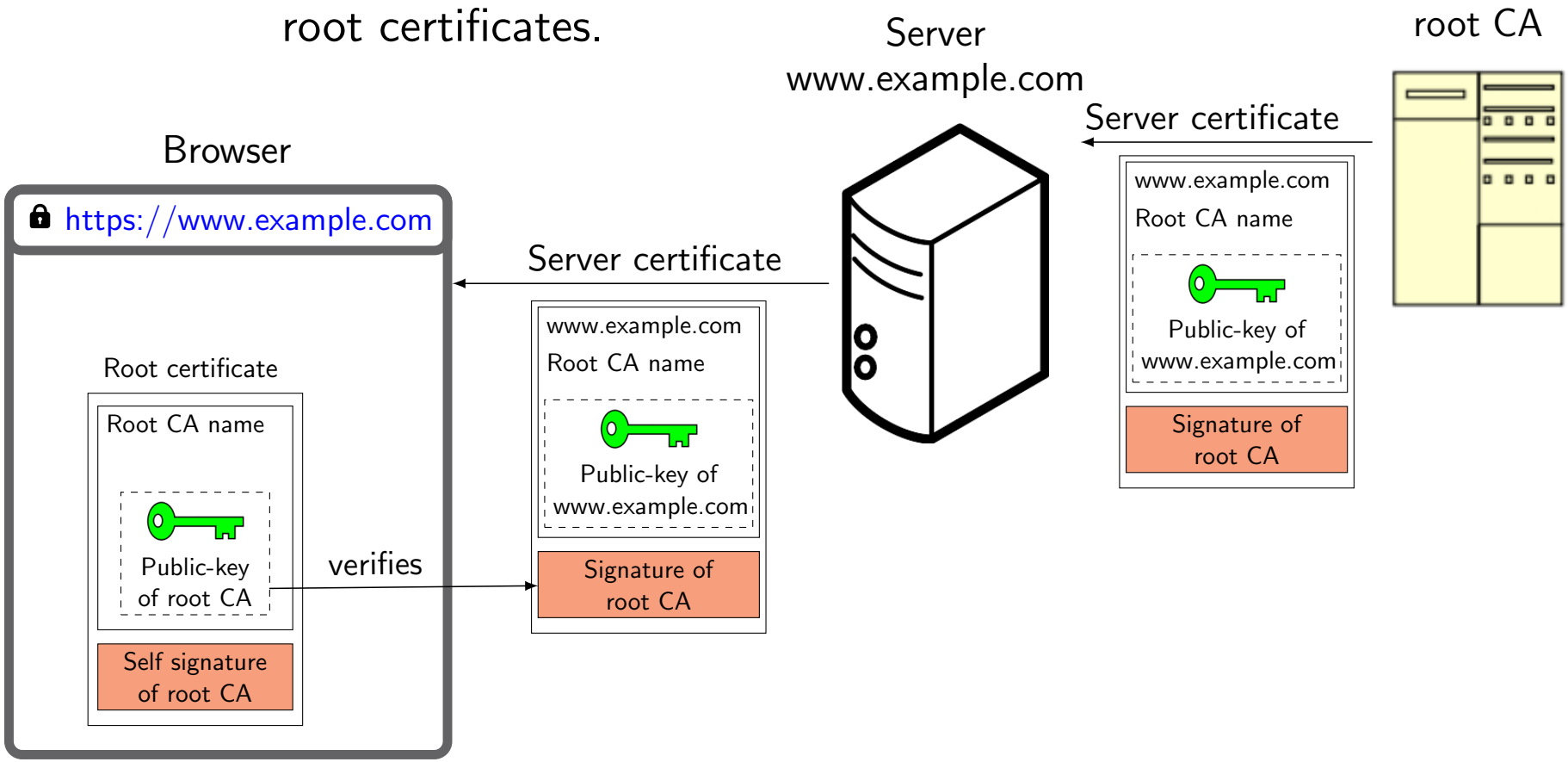
Server authentication in HTTPS

- CA issues a certificate to the server to authenticate the server's public-key
  - The server expects the CA's certificate to be contained in most clients web browser.
  - One needs to trust the browser's publisher to include correct root certificates.



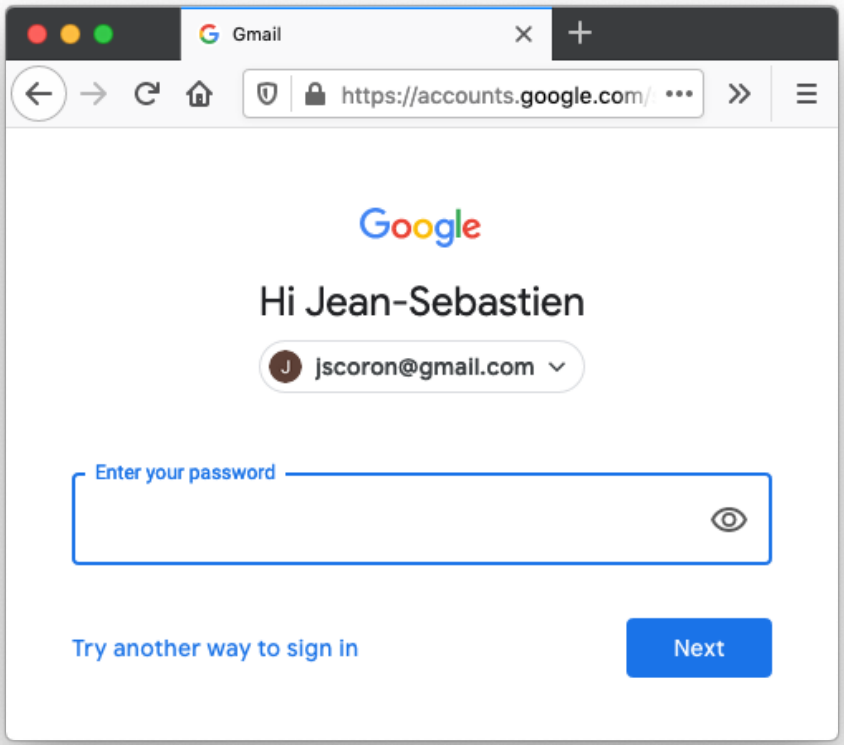
Server authentication in HTTPS

- CA issues a certificate to the server to authenticate the server's public-key
  - The server expects the CA's certificate to be contained in most clients web browser.
  - One needs to trust the browser's publisher to include correct root certificates.



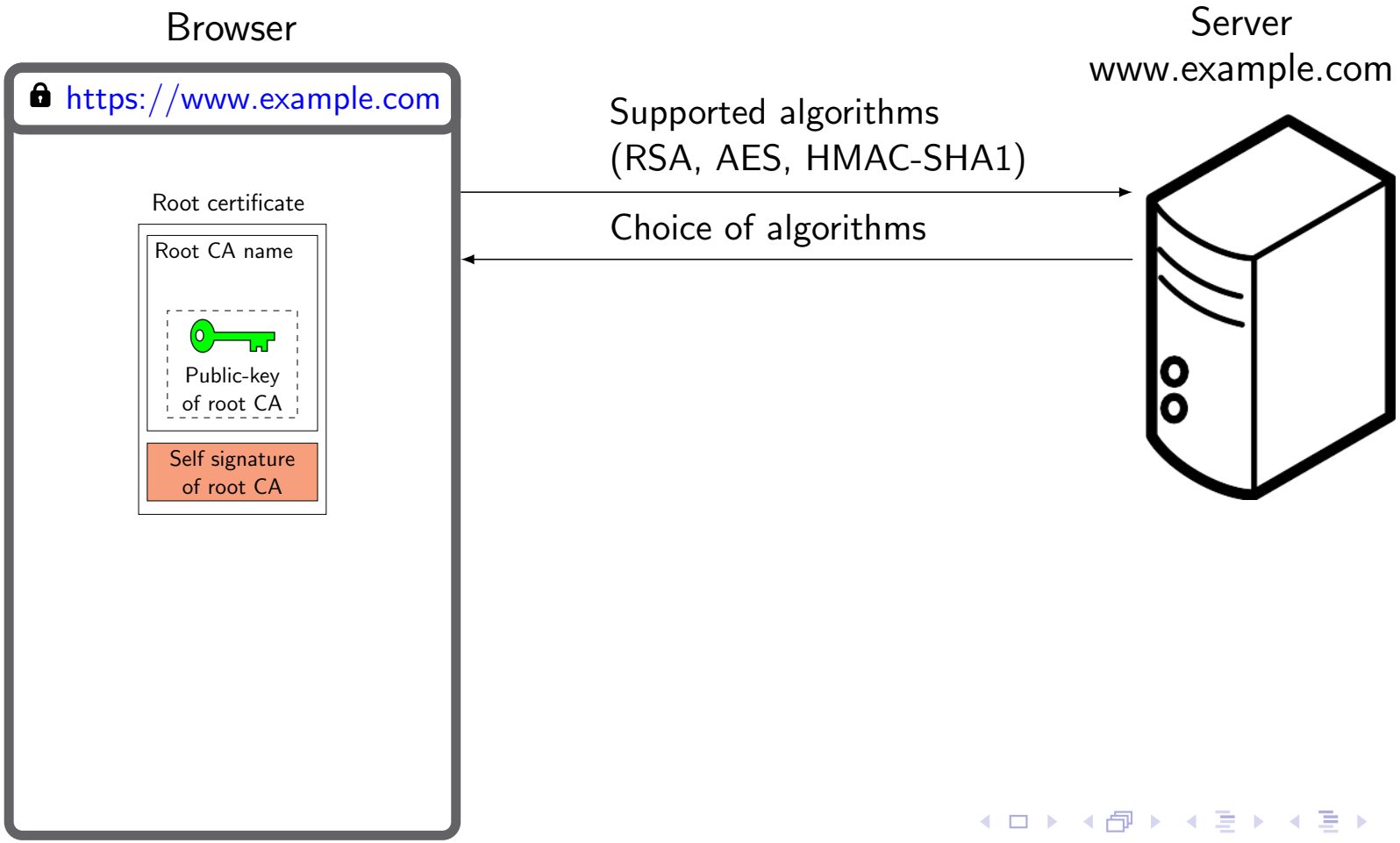
Client authentication in HTTPS

- In a typical HTTPS connection, TLS authenticates the server.
- A website can authenticate the user at the application layer:
  - For example, using a password or a passkey over TLS.
- TLS can also authenticate the client with a client certificate:
  - This is called mutual TLS (mTLS).
  - Used, for example, between services and in enterprise systems.



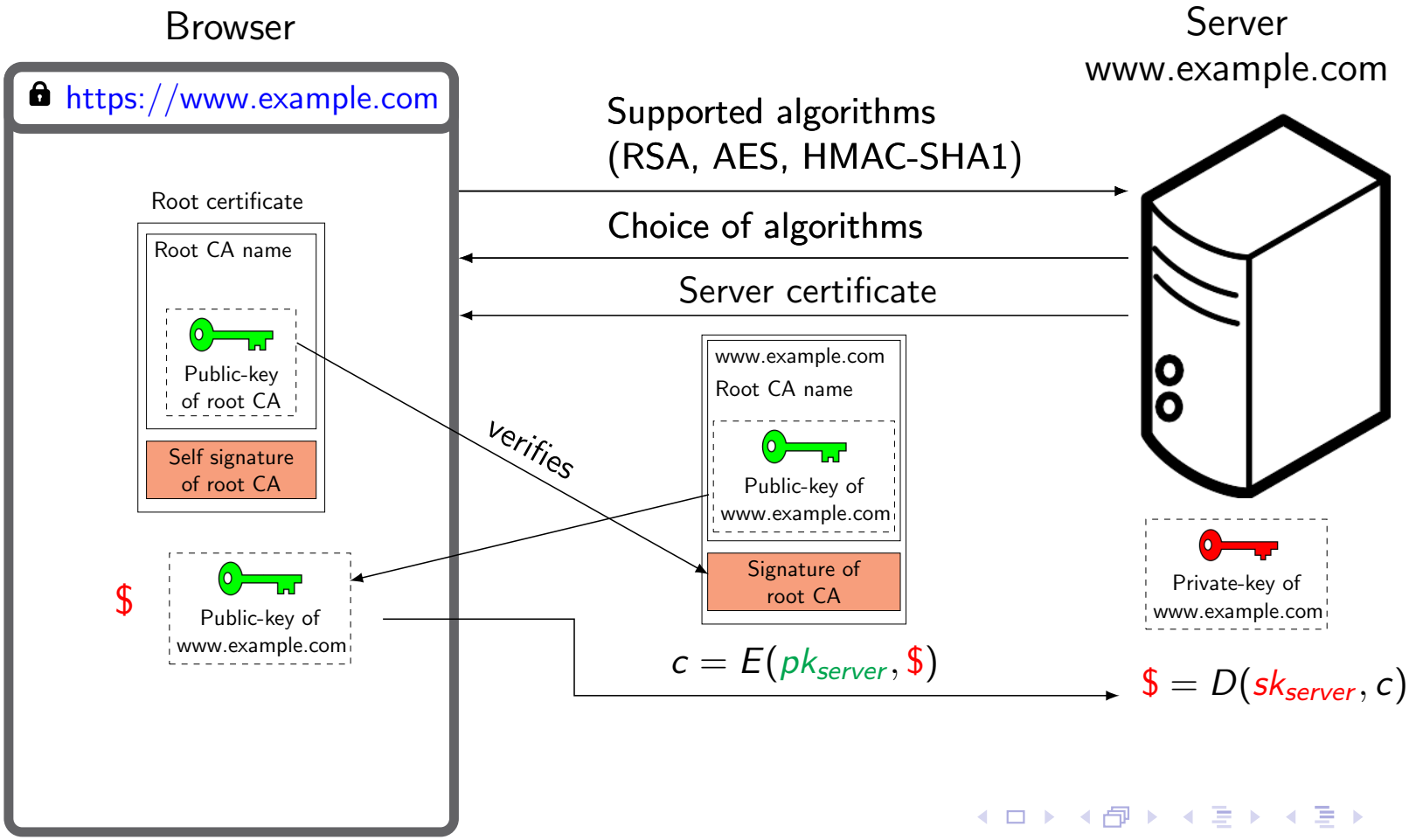
Historical RSA key transport in TLS 1.2 and earlier

- Historical example: RSA key transport (not used in TLS 1.3)
  - Negotiation for algorithms used.
  - Certificate verification and RSA encryption of a premaster secret.
  - Symmetric encryption for traffic encryption.



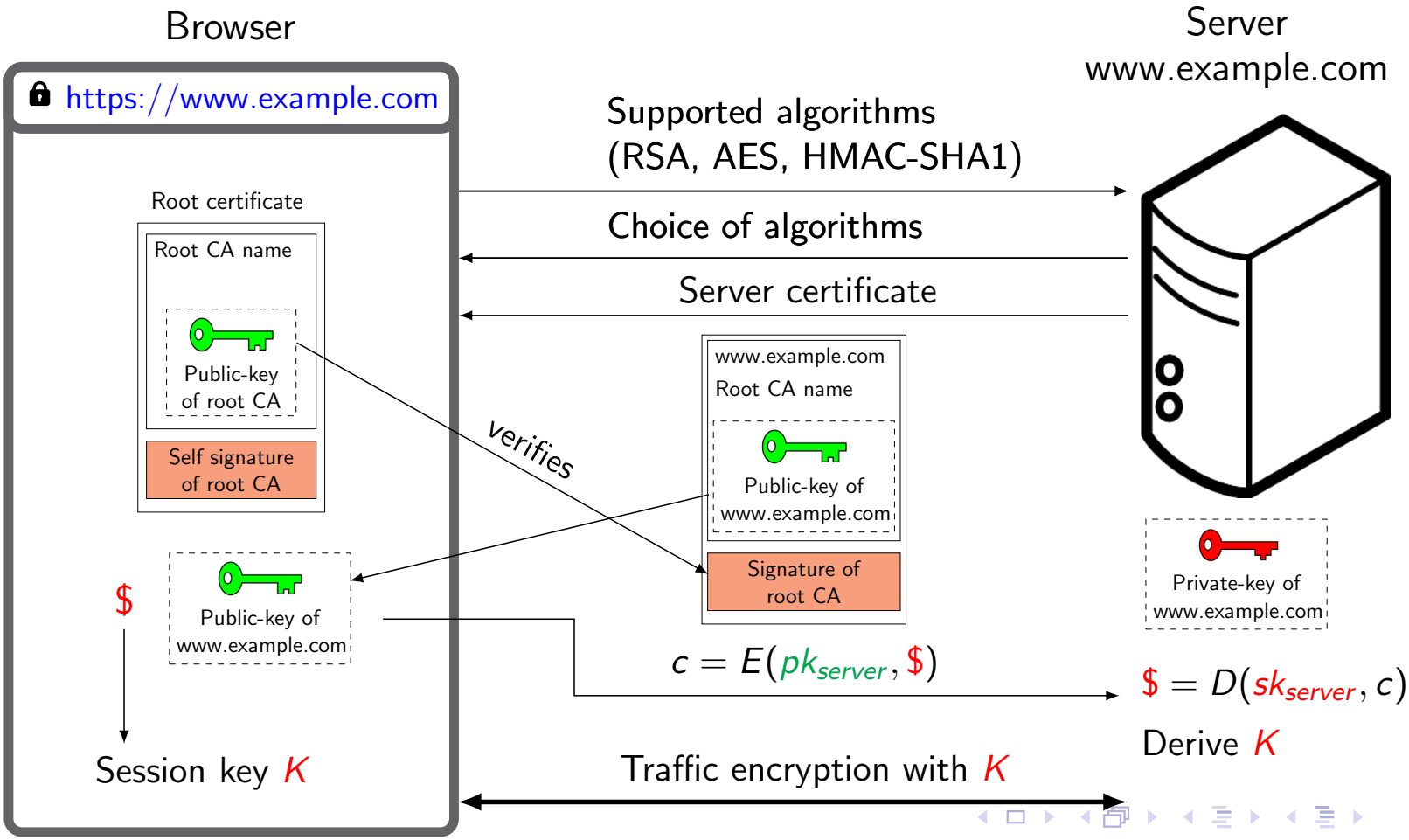
Historical RSA key transport in TLS 1.2 and earlier

- Historical example: RSA key transport (not used in TLS 1.3)
  - Negotiation for algorithms used.
  - Certificate verification and RSA encryption of a premaster secret.
  - Symmetric encryption for traffic encryption.

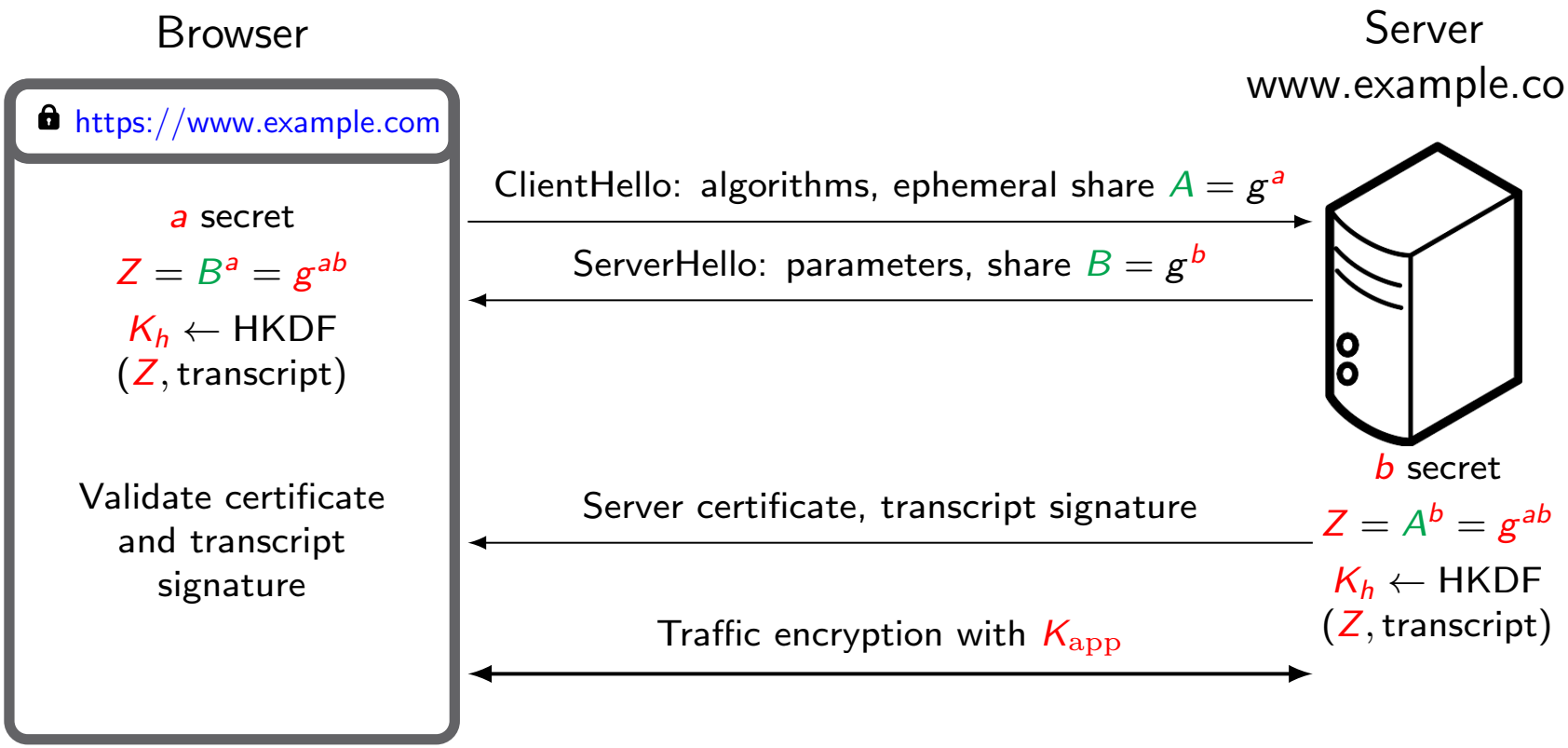


Historical RSA key transport in TLS 1.2 and earlier

- Historical example: RSA key transport (not used in TLS 1.3)
  - Negotiation for algorithms used.
  - Certificate verification and RSA encryption of a premaster secret.
  - Symmetric encryption for traffic encryption.

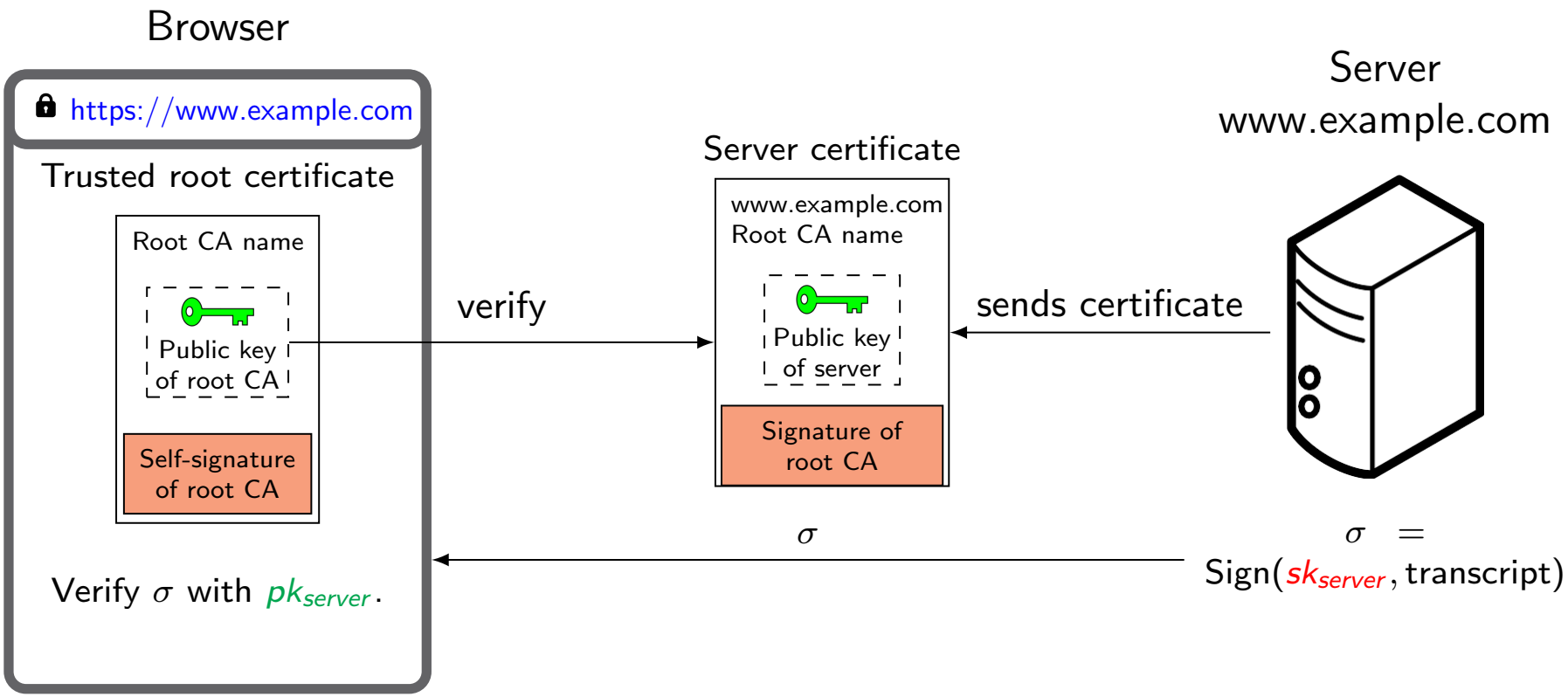


TLS 1.3: certificate-authenticated handshake



- K_{app} is derived separately from K_h; key derivation is schematic.
- Certificate and signature authenticate the server (see next slide).
- For elliptic curves, shares are $A = aG$ and $B = bG$.
- Ephemeral DH provides forward secrecy if ephemeral secrets are erased; historical RSA key transport does not.

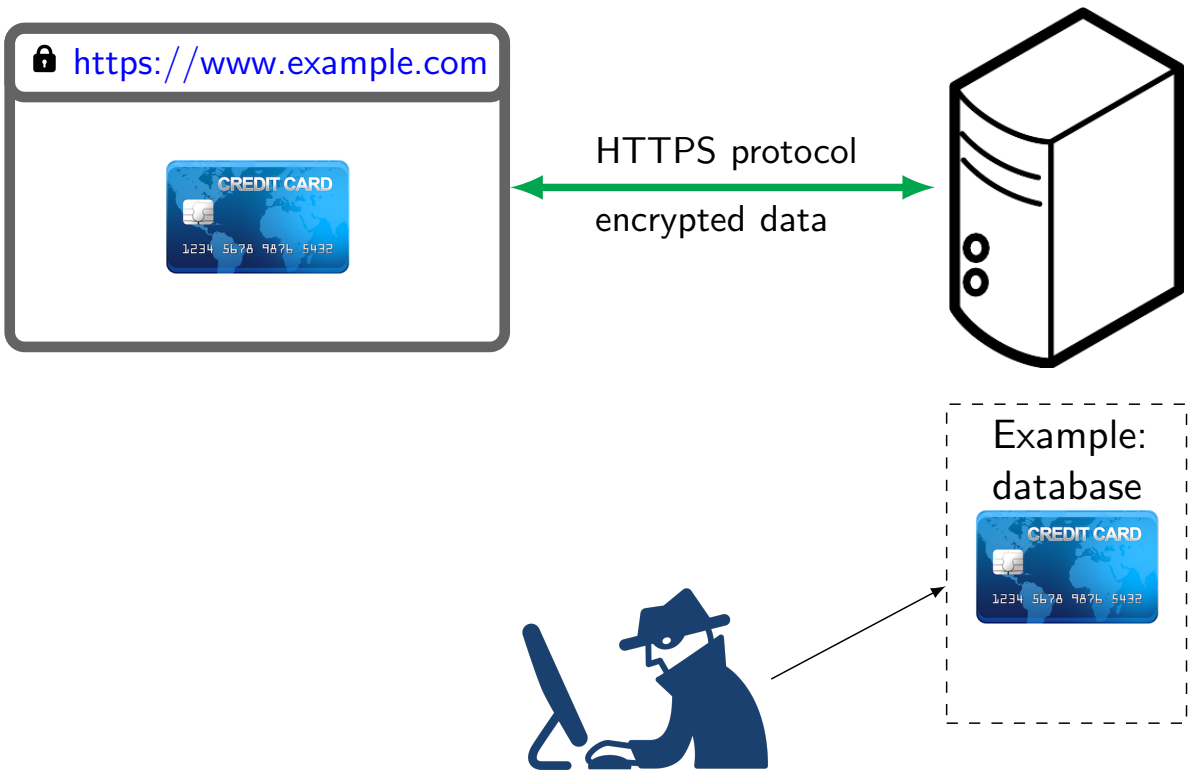
TLS 1.3: how the certificate authenticates the server



- The browser checks the hostname and certificate validity.
- Certificate and signature are encrypted with handshake keys.
- The certificate binds the name to pk_{server}; the signature proves possession of sk_{server} and binds the handshake.

Credit card via HTTPS

- HTTPS protects the credit card number in transit between the browser and the TLS server.
  - It does not protect against compromised endpoints or a malicious website.
  - If card data is stored, an attacker may target the storage system.



Conclusion

- Security models specify the adversary's goal and capabilities.
  - IND-CCA2 captures confidentiality against chosen-ciphertext attacks; unforgeability captures signature security.
- Textbook RSA is insecure; use secure schemes such as RSA-OAEP and RSA-PSS.
  - Hybrid encryption combines public-key encryption with authenticated symmetric encryption.
- Certificates bind identities to public keys through trusted certification authorities.
- TLS combines authentication, key exchange and symmetric encryption to protect communications.
  - Security also depends on certificate validation, implementations and endpoints.