CSE 3400/CSE 5850 - Introduction to Computer & Network Security
/ Introduction to Cybersecurity

Lecture 11
Public Key Cryptography– Part II

Ghada Almashaqbeh
UConn

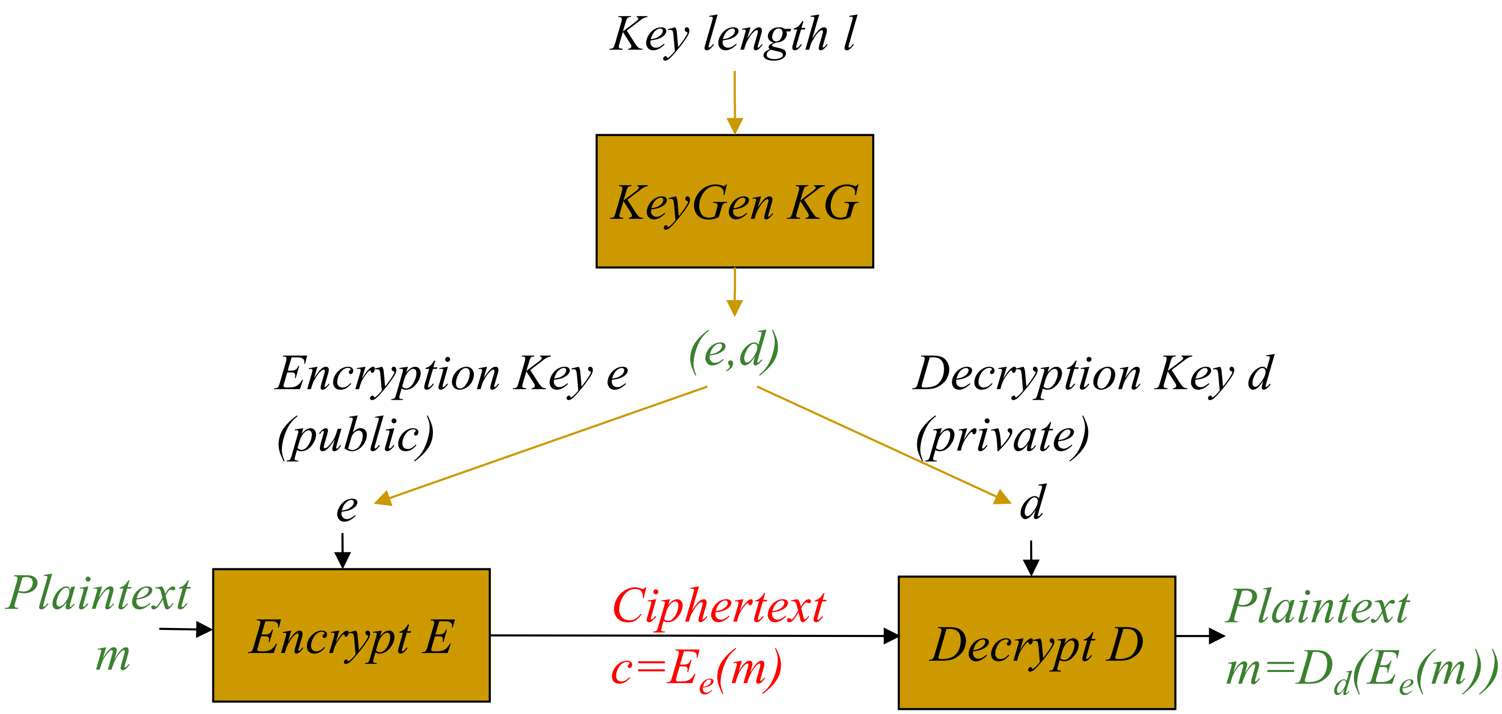
*Adapted from the textbook slides

Outline

- Public key encryption.
- Digital signatures.

Public Key Encryption

Public Key Encryption



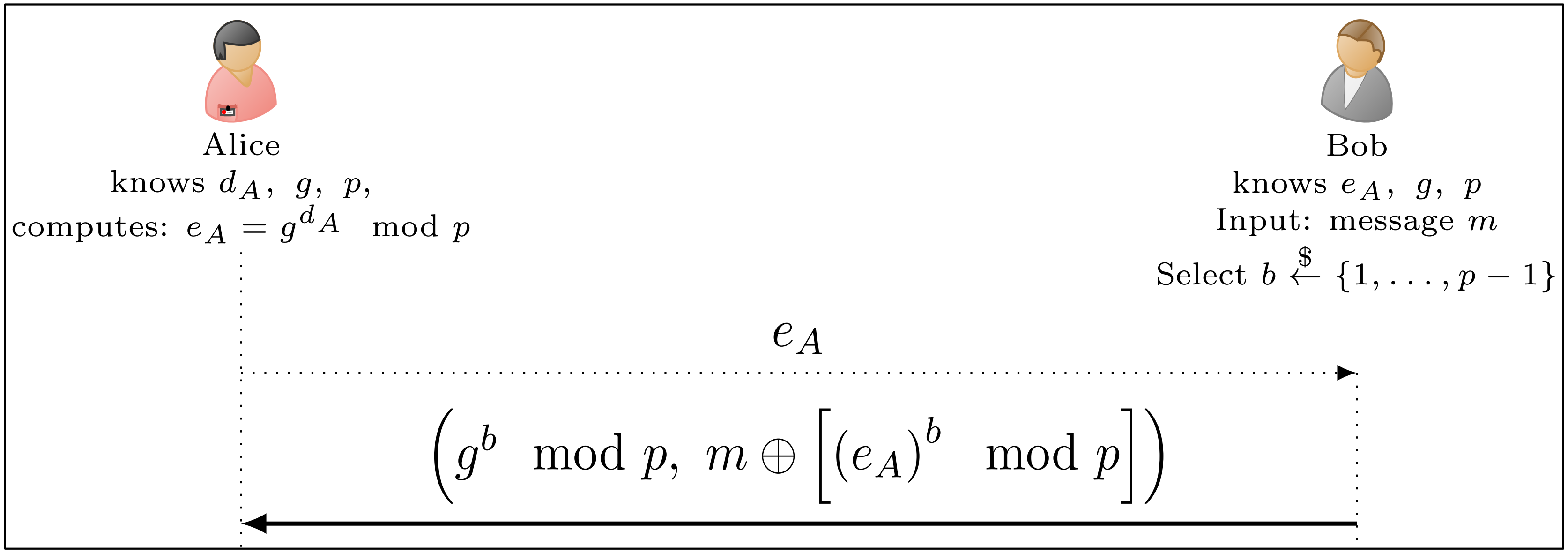
Public Key Encryption IND-CPA Security

- Same security game as before.
  - The attacker chooses two messages of the same length, and is challenged to correctly guess which of these messages was encrypted by the challenger.
- The difference is that the attacker does not need an oracle access to the encryption oracle!
  - The public encryption key (but not the private decryption key) is known to everyone, including the adversary, and can use it to encrypt any message he wants.
- Can a deterministic public key encryption scheme be an IND-CPA secure?

Discrete Log-based Encryption

- We will study two constructions:
  - An adaptation of DH key exchange protocol to perform encryption.
  - ElGamal encryption scheme.

The DH Encryption Scheme



Encryption:

$$E_{e_A}(m) = \left\{ \begin{array}{l} b \xleftarrow{\$} \{1, \ldots, p-1\} \\ \text{Return } \left(g^b \mod p\ ,\ m \oplus (e_A)^b \mod p\right) \end{array} \right\}$$

Notice that *the ciphertext is the pair* $\left(g^b \mod p\ ,\ m \oplus (e_A)^b \mod p\right)$.

Decryption:

$$D_{d_A}(c_b, c_m) = c_m \oplus \left[(c_b)^{d_A} \mod p\right]$$

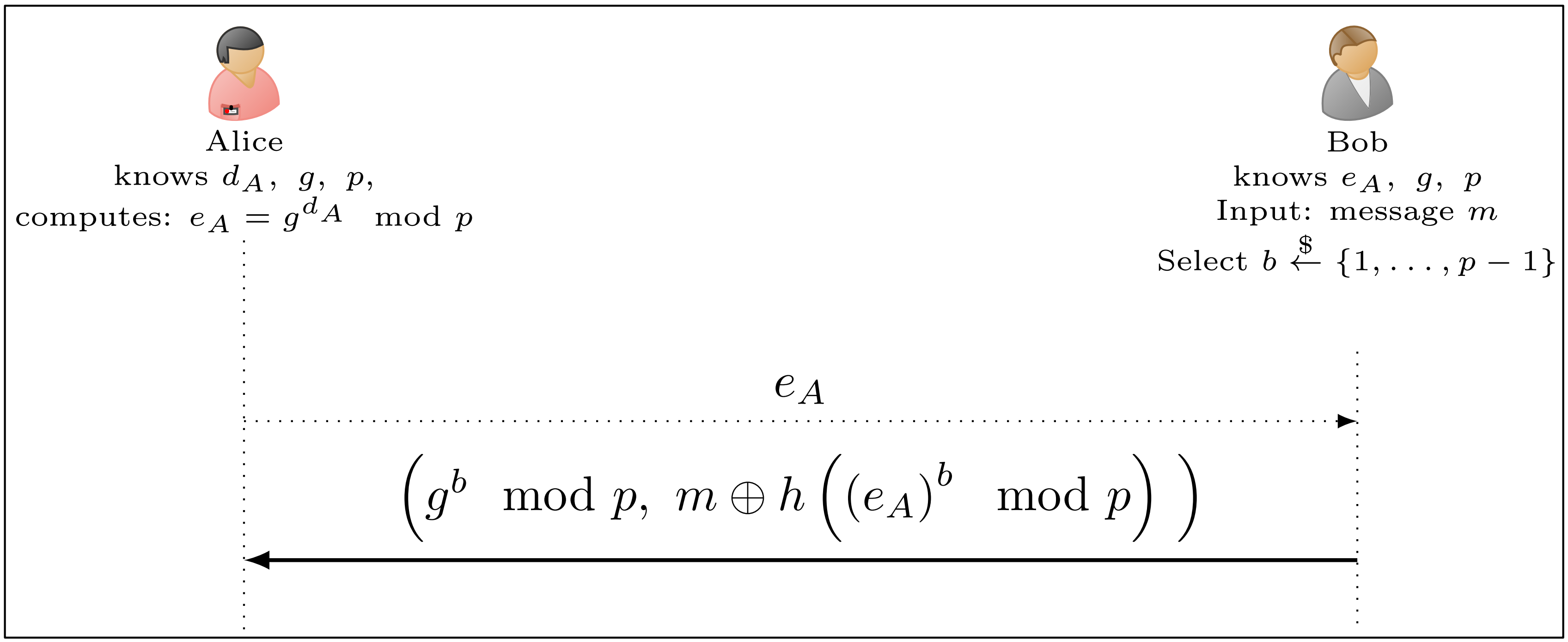
The DH Encryption Scheme---Correctness and Security

$$\begin{aligned} D_{d_A}(E_{e_A}(m)) &= D_{d_A}\left(g^b \mod p,\ m \oplus (e_A)^b \mod p\right) \\ &= D_{d_A}\left(g^b \mod p,\ m \oplus \left(g^{d_A} \mod p\right)^b \mod p\right) \\ &= \left(m \oplus \left(g^{d_A} \mod p\right)^b \mod p\right) \oplus \left[\left(g^b \mod p\right)^{d_A} \mod p\right] \\ &= m \oplus \left(g^{d_A \cdot b} \mod p\right) \oplus \left(g^{b \cdot d_A} \mod p\right) \\ &= m \end{aligned}$$

- May not be secure!
  - Believed to be secure under the CDH assumption, however, it is not always true! g^{ab} may leak some information (or bits) as we studied before.
- Solution?
  - The hashed DH encryption scheme.

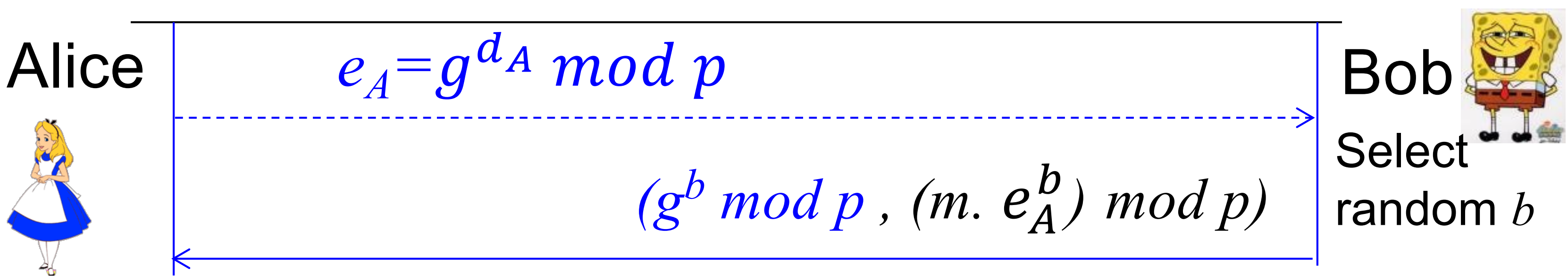
The Hashed DH Encryption Scheme

- Secure if $h(g^{b \cdot d_A} \bmod p)$ is pseudorandom (so the hash function must be a randomness-extractor hash function).



ElGamal Public Key Encryption

- To encrypt message m to Alice, whose public key is $e_A = g^{d_A} \bmod p$:
  - Bob selects random b
  - Sends: $g^b \bmod p$, $m.(e_A)^b = m.g^{b \cdot d_A} \bmod p$



ElGamal Public Key Encryption

- Encryption:

$$E^{EG}_{e_A}(m) \leftarrow \left\{ \left(g^b \mod p \ , \ m \cdot e_A^b \mod p \right) | b \xleftarrow{\$} [2, p-1] \right\}$$

- Decryption:

$$D_{d_A}(x, y) = x^{-d_A} \cdot y \mod p$$

- Correctness:

$$\begin{aligned} D_{d_A}(g^b \mod p \ , \quad m \cdot \quad e_A^b \mod p) \ &= \\ &= \left[\left(g^b \mod p \right)^{-d_A} \cdot \left(m \cdot \left(g^{d_A} \right)^b \mod p \right) \right] \mod p \\ &= \left[g^{-b \cdot d_A} \cdot m \cdot g^{b \cdot d_A} \right] \mod p \\ &= m \end{aligned}$$

ElGamal Public Key Cryptosystem

- Problem: $g^{b \cdot d_A} \ mod\, p$ may leak bit(s)…
- \`Classical' DH solution: securely derive a key: $h\left(g^{a_i b_i} mod\ p\right)$
- El-Gamal's solution: use a group where DDH believed to hold
  - Note: message must be encoded as member of the group!
- What is special about ElGamal Encryption?
  - Homomorphism!

ElGamal PKC: homomorphism

- Multiplying two ciphertexts produces a ciphertext of the multiplication of the two plaintexts.
- Given two ciphertexts:
  - $E_{e_A}(m_1) = (x_1, y_1) = (g^{b_1} \mod p, m_1 \, . \, g^{b_1 \cdot d_A} \mod p)$
  - $E_{e_A}(m_2) = (x_2, y_2) = (g^{b_2} \mod p, m_2 \, . \, g^{b_2 \cdot d_A} \mod p)$
- $Mult\big((x_1, y_1), (x_2, y_2)\big) \equiv (x_1 x_2, y_1 y_2)$
- Homomorphism:
- $= (g^{b_1+b_2} \mod p, m_1 \cdot m_2 . \, g^{(b_1+b_2) \cdot d_A} \mod p) =$
  $= E_{e_A}(m_1 \cdot m_2)$
- ➔ compute $E_{e_A}(m_1 \cdot m_2)$ from $E_{e_A}(m_1)$, $E_{e_A}(m_1)$

RSA Public Key Encryption

- First proposed – and still widely used
- Select two large primes p,q ; let $n=pq$
- Select prime e (public key: $<n,e>$)
  - Or co-prime with $\Phi(n) = (p-1)(q-1)$
- Let private key be $d=e^{-1} \bmod \Phi(n)$ (i.e., $ed=1 \bmod \Phi(n)$)
- Encryption: $RSA.E_{e,n}(m)=m^e \bmod n$
- Decryption: $RSA.D_{d,n}(c)=c^d \bmod n$
- Correctness: $D_{d,n}(E_{e,n}(m)) = (m^e)^d = m^{ed} = m \bmod n$
  - Intuitively: $ed=1 \bmod \Phi(n) \rightarrow m^{ed} = m \bmod n$
- But why? Remember Euler's theorem.

RSA Public Key Cryptosystem

- Correctness: $D_{d,n}(E_{e,n}(m))= m^{ed} \; mod \; n$
  - $m^{ed}=m^{ed}= m^{1+l\,\Phi(n)} =m\, m^{l\,\Phi(n)} =m\, (m^{\Phi(n)})^{l}$
  - $m^{ed} \, mod\, n =m\, (m^{\Phi(n)}\, mod\, n)^{l}\, mod\, n$
  - Euler's Theorem: $m^{\Phi(n)}\, mod\, n=1\, mod\, n$
  - $\Rightarrow D_{d,n}(E_{e,n}(m))= m^{ed}\, mod\, n=m\, 1^{l}\, mod\, n =m$
- Comments:
  - $m<n \Rightarrow m= m\, mod\, n$
  - Euler's Theorem holds (only) if m, n are co-primes
  - If not co-primes? Use Chinese Reminder Theorem
    - A nice, not very complex argument
    - But: beyond our scope – take Crypto!
  - Number of messages co-prime to n ?!

The RSA Problem and Assumption

- RSA problem: Find m , given (n,e) and 'ciphertext' value $c=m^e \bmod n$
- RSA assumption: if (n,e) are chosen `correctly', then the RSA problem is `hard'
  - I.e., no efficient algorithm can find m with non-negligible probability
  - For `large' n and $m \overset{\$}{\leftarrow} \{1, \dots, n\}$
- Relation between RSA and factoring:
  - Factoring algorithm ➔ algorithm to 'break' RSA
    - Simply use that to find the factors of n, then $\Phi(n)$, then compute the decryption key so you can reveal m.
  - But: RSA-breaking may *not* allow factoring

RSA PKC Security

- It is a deterministic encryption scheme $\rightarrow$ cannot be IND-CPA secure.
- RSA assumption does not rule out exposure of partial information about the plaintext.

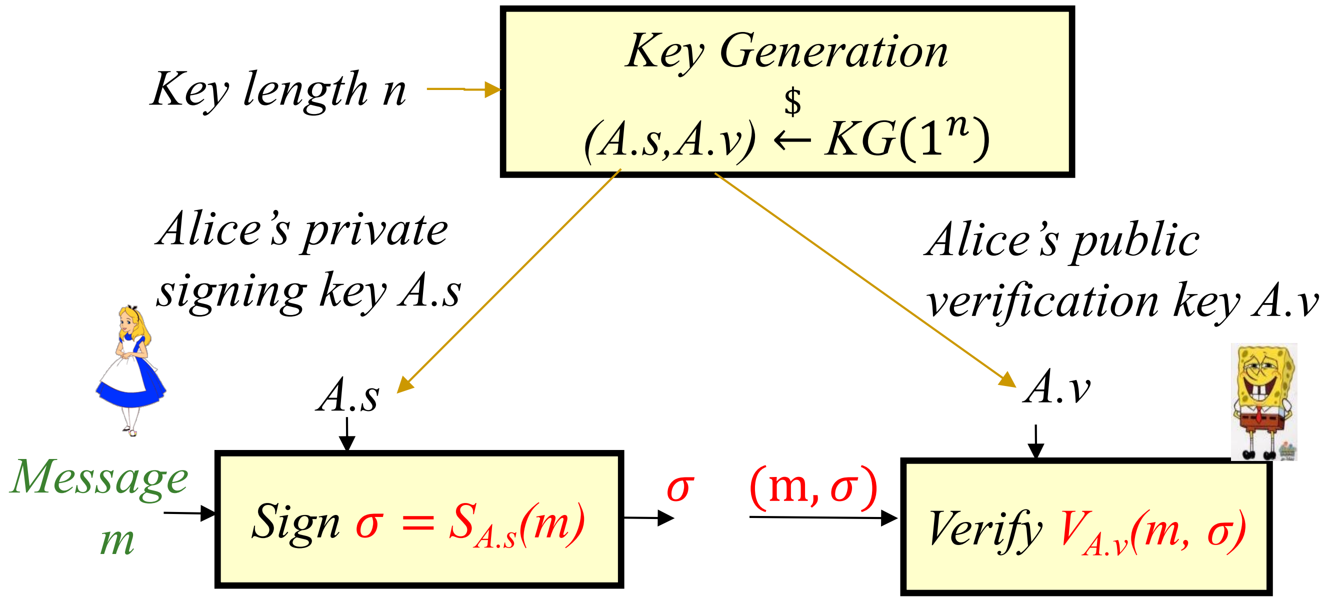
A solution: apply a random padding to the plaintext then encrypt using RSA.

Padded RSA

- Pad and Unpad functions: $m = Unpad(Pad(m;r))$
  - Encryption with padding: $c = [Pad(m,r)]^e \bmod n,$
  - Decryption with unpad: $m = Unpad(c^d \bmod n)$
- So it adds randomization to Prevent detection of repeating plaintext
- Padding must be done carefully; certain padding algorithms still do not guarantee CPA security.

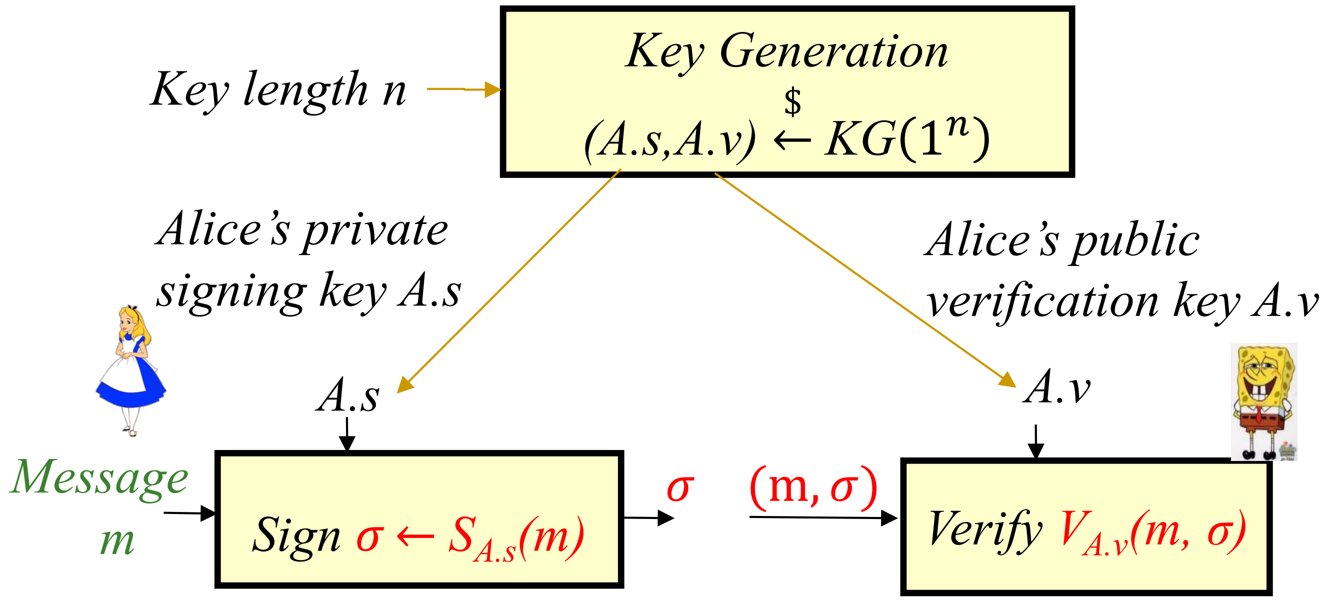
Digital Signature

Public Key Digital Signatures



- Sign using a private, secret signing key ($A.s$ for Alice)
- Validate using a <u>public</u> verification key ($A.v$ for Alice)
- Everybody can validate signatures at any time
  - Provides authentication, integrity **<u>and</u>** evidence / non-repudiation
  - MAC: 'just' authentication + integrity, no evidence, can repudiate

Digital Signatures Security: Unforgeability



- Unforgeability: given v, attacker should be unable to find **any** 'valid' (m, σ), i.e., $V_v(m, \sigma)=OK$
  - Even when attacker can select messages m', receive $\sigma'=S_s(m')$ *– so it has access to the signing oracle*
  - And the forgery is for a new message (that was not asked to the oracle).

Digital Signature Scheme Definition

Definition 1.4 (Signature scheme and its correctness). *A* signature scheme *is defined by a tuple of three efficient (PPT) algorithms,* $\mathcal{S} = (\mathcal{KG}, \mathcal{Sign}, \mathcal{Verify})$, *and a set M of messages, such that:*

$\mathcal{KG}$ *is a randomized algorithm that maps a unary string (security parameter 1^n) to a pair of binary strings* *(s, v), the signing and verification keys, respectively.*

$\mathcal{Sign}$ *is an algorithm[8] that receives two binary strings as input, a signing key $s \in \{0,1\}^*$ and a message $m \in M$, and outputs another binary string $\sigma \in \{0,1\}^*$. We call σ the* signature *of m using signing key s.*

$\mathcal{Verify}$ *is a predicate that receives three binary strings as input: a verification key v, a message m, and σ, a purported signature over m. $\mathcal{Verify}$ should output* True *if σ is the signature of m using s, where s is the signature key corresponding to v (generated with v).*

Usually, M is a set of binary strings of some length. If M is not defined, then this means that any binary string may be input, i.e., the same as $M = \{0,1\}^*$.

We say that a signature scheme $(\mathcal{KG}, \mathcal{Sign}, \mathcal{Verify})$ is correct, *if for every security parameter 1^n holds:*

$$\left(\forall (s,v) \xleftarrow{\$} \mathcal{KG}(1^n),\ m \in M\right) \mathcal{Verify}_v(m, \mathcal{Sign}_s(m)) = \text{‘}Ok\text{’} \qquad (1.31)$$

Digital Signature Scheme Security

Algorithm 1 The existentially unforgeable game $EUF_{\mathcal{A},\mathcal{S}}(1^n)$ between signature scheme $\mathcal{S} = (\mathcal{KG}, \mathcal{Sign}, \mathcal{Verify})$ and adversary $\mathcal{A}$.

$(s, v) \xleftarrow{\$} \mathcal{S}.\mathcal{KG}(1^n)$
$(m, \sigma) \xleftarrow{\$} \mathcal{A}^{\mathcal{S}.\mathcal{Sign}_s(\cdot)}(v, 1^n)$
return $(\mathcal{S}.\mathcal{Verify}_v(m, \sigma) \wedge (\mathcal{A}$ didn't give m as input to $\mathcal{S}.\mathcal{Sign}_s(\cdot)))$

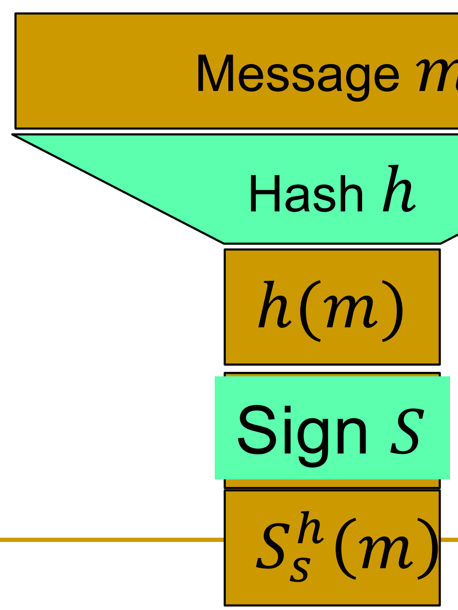
Definition 1.4. *The* existential unforgeability advantage function *of adversary $\mathcal{A}$ against signature scheme $\mathcal{S}$ is defined as:*

$$\varepsilon^{EUF}_{\mathcal{S},\mathcal{A}}(1^n) \equiv \Pr\left(EUF_{\mathcal{A},\mathcal{S}}(1^n) = \text{TRUE}\right) \tag{1.3}$$

Where the probability is taken over the random coin tosses of $\mathcal{A}$ and of $\mathcal{S}$ during the run of $EUF_{\mathcal{A},\mathcal{S}}(1^n)$ with input (security parameter) 1^n, and $EUF_{\mathcal{A},\mathcal{S}}(1^n)$ is the game defined in Algorithm 1.

RSA Signatures

- Secret signing key s, public verification key v
- σ = RSA.$S_s(m) = m^s \bmod n$,
  RSA.$V_v(m, \sigma) = \{ OK \text{ if } m = \sigma^v \bmod n; \text{ else, } FAIL \}$
- Long messages?
  - Hint: use collision resistant hash function (CRHF)
  - σ = RSA.$S_s(m) = h(m)^s \bmod n$,
    RSA.$V_v(m, \sigma) = \{ OK \text{ if } h(m) = \sigma^v \bmod n; \text{ else, } FAIL \}$



Discrete-Log Digital Signature?

- Can we sign based on assuming discrete log is hard?
- Most well-known, popular scheme: DSA
  - Digital Signature Algorithm, by NSA/NIST
  - Details: crypto course

Covered Material From the Textbook

- Chapter 1: Section: 1.4
- Chapter 6:
  - Sections 6.4 (except 6.4.4)
  - Section 6.5 (except 6.5.6, 6.5.7, and 6.5.8),
  - And Section 6.6 (except RSA with message recovery and appendix)

Thank You!