Cryptography

Lecture 11: RSA Assumption & Public-Key Encryption Schemes

January 7, 2025

Overview

Contents

1. Classical cryptography
   (Shift & Vigenère cipher, one-time pad, perfect secrecy)
2. Security definitions & threat models
   (Computational security, CPA & CCA)
3. Private-key cryptography
   (Message authentication, hash functions, primitives, relevant ciphers)
4. Public-key cryptography
   (Assumptions, key management, digital signatures, relevant ciphers)

Foundations

(aka Fermat's little theorem, which was first proven by Euler)

Theorem (§9.13 Euler's theorem)

Let $\mathbb{G} = (G, \cdot, 1)$ be finite commutative group and $\ell = |G|$.
Then $g^\ell = 1$ for every $g \in G$.

Consequences

- $g^a = g^{a \bmod \ell}$ for all $g \in G$ and $a \in \mathbb{Z}$
  (compute modulo group order in exponents)
- $f_a \colon G \to G$ with $f_a(g) = g^a$ for all $g \in G$ is bijection
  for all $a \in \mathbb{N}_+$ such that $\gcd(a, \ell) = 1$
  (every group element is a-th power; i.e. has unique a-th root)
- As usual given $g \in G$ and $a \in \mathbb{N}_+$ with $\gcd(a, \ell) = 1$
  we write $g^{\frac{1}{a}}$ or $\sqrt[a]{g}$ for unique $g' \in G$ such that $(g')^a = g$

Example

- $\mathbb{Z}_{10} = (\{0, \ldots, 9\}, \cdot, 1)$ is monoid with $|\mathbb{Z}_{10}| = 10$ elements
  (no group; computation with usual multiplication modulo 10)
- $\mathbb{Z}_{10}^* = (\{1, 3, 7, 9\}, \cdot, 1)$ is subgroup of $\mathbb{Z}_{10}$ of order $|\mathbb{Z}_{10}^*| = 4$
  (contains only those elements of $\mathbb{Z}_{10}$ that are invertable,
  but computation is still with usual multiplication modulo 10)
- Element $a \in \mathbb{Z}_\ell$ is invertable (in $\mathbb{Z}_\ell$) if and only if $\gcd(a, \ell) = 1$
  ($5 \notin \mathbb{Z}_{10}^*$ and $7 \in \mathbb{Z}_{10}^*$ because $\gcd(5, 10) = 5 \neq 1$ and $\gcd(7, 10) = 1$)

Example (cont'd)

- All the following calculations will be inside group $\mathbb{Z}_{10}^*$ of order 4
- Let $a = 2027 = 3 \mod 4$ permitted by $\gcd(2027, 4) = 1$

computation in exponent modulo $\phi(10) = 4$

$$f_a(1) = 1^{2027} = 1^{2027 \mod 4} = 1^3 = 1 \mod 10$$

$$f_a(3) = 3^{2027} = 3^{2027 \mod 4} = 3^3 = 27 = 7 \mod 10$$

$$f_a(7) = 7^{2027} = 7^{2027 \mod 4} = 7^3 = 49 \cdot 7 = 9 \cdot 7 = 63 = 3 \mod 10$$

$$f_a(9) = 9^{2027} = 9^{2027 \mod 4} = \underbrace{9^3 = 81 \cdot 9 = 1 \cdot 9 = 9 \mod 10}$$

computation on group elements modulo 10

- Hence

$$\sqrt[2027]{1} = 1 \qquad \sqrt[2027]{7} = 3 \qquad \sqrt[2027]{3} = 7 \qquad \sqrt[2027]{9} = 9$$

Foundations

(We also write $\mathbb{Z}_\ell^*$ for set Z_ℓ^*; i.e. identify structure with carrier)

Motivation

- Let p, p' be different primes, $\ell = pp'$, and $e > 2$ with $\gcd\big(e, \phi(\ell)\big) = 1$
- We compute in commutative group $\mathbb{Z}_\ell^*$ of order $\phi(\ell)$ (Lemma §9.12)
- Easy to compute m^e for every $m \in \mathbb{Z}_\ell^*$ by iterated squaring
- Seems hard to compute $c^{\frac{1}{e}} = \sqrt[e]{c}$ given only ℓ, e, and $c \in \mathbb{Z}_\ell^*$

Leonhard Euler (* 1707; † 1783)

- Swiss mathematician & physicist
- Responsible for most of mathematical notation
- Polymath

Rivest-Shamir-Adleman (RSA)

Adi Shamir (* 1952)

- Israelian computer scientist
- Professor at Weizmann institute & ENS Paris
- Co-inventor of differential cryptanalysis

© The Royal Society

Leonard Adleman (* 1945)

- US computer scientist
- Professor at University of Southern California
- Coined “computer virus” & founded DNA computing

RSA Assumption

§10.1 Definition (RSA key generation)

RSA key generator is PPT algorithm that given security parameter 1^n with $n \in \mathbb{N}$ returns (ℓ, e, d) such that

- there exist primes $p, p' \in \mathbb{N}$ such that $2^{n-1} < p < p' < 2^n$ and $\ell = pp'$
- $e, d \in \mathbb{Z}^*_{\phi(\ell)}$ such that $e \cdot d = 1 \bmod \phi(\ell)$

Typical implementation

1. Generate uniform n-bit primes $p \neq p'$ and compute $\ell = pp'$
   (see Algorithm 8.34 in [Katz & Lindell] with AKS primality test)
2. Select $e \in \mathbb{Z}^*_{\phi(\ell)}$ (i.e. $e \in \mathbb{N}$ such that $\gcd\big(e, \phi(\ell)\big) = 1$)
   (typically $e = 3$ which requires $\gcd(p, 3) = 1 = \gcd(p', 3)$)
3. Compute $d = e^{-1}$ in $\mathbb{Z}^*_{\phi(\ell)}$ and return (ℓ, e, d)

RSA Assumption

RSA key generator f

Alice

$(\ell, e, d) \leftarrow f(1^n)$

uniform $c \in \mathbb{Z}_\ell^*$

Eve

RSA Assumption

RSA key generator f

Alice

$(\ell, e, d) \leftarrow f(1^n)$

uniform $c \in \mathbb{Z}_\ell^*$

Challenge (ℓ, e, c)

Eve

RSA Assumption

RSA key generator f

Alice
$(\ell, e, d) \leftarrow f(1^n)$
uniform $c \in \mathbb{Z}_\ell^*$

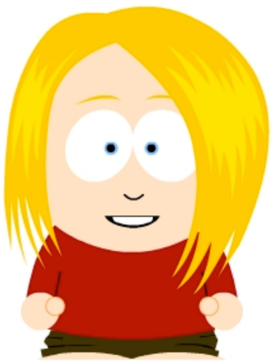
Challenge (ℓ, e, c)

Candidate m

Eve
Determines m

RSA Assumption

RSA key generator f



Alice

$(\ell, e, d) \leftarrow f(1^n)$

uniform $c \in \mathbb{Z}_\ell^*$

$c' = m^e \bmod \ell$

Return $c \stackrel{?}{=} c'$

Challenge (ℓ, e, c)

Candidate m

Eve

Determines m

RSA Assumption

§10.2 Definition (RSA game)

Let $n \in \mathbb{N}$, f RSA key generator and $\mathcal{A}$ stateful algorithm.
RSA game $\mathrm{RSA}_{f,\mathcal{A}}(n)$ is

1. $(\ell, e, d) \leftarrow f(1^n)$
2. Select $c \in \mathbb{Z}_\ell^*$ uniformly and send (ℓ, e, c) to adversary $\mathcal{A}$
3. $\mathcal{A}$ selects and returns element $m \in \mathbb{Z}_\ell^*$
4. $c' = m^e \bmod \ell$
5. Return 1 if $c = c'$ ($\mathcal{A}$ wins) and return 0 otherwise

Notes

- $\mathcal{A}$ receives (ℓ, e, c) and is expected to compute $\sqrt[e]{c} \bmod \ell$ (in $\mathbb{Z}_\ell^*$)
- Verify $m \stackrel{?}{=} \sqrt[e]{c} \bmod \ell$ by checking $m^e \stackrel{?}{=} c \bmod \ell$

RSA Assumption

§10.3 Definition (RSA assumption)

RSA problem is hard for RSA key generator f if for every PPT algorithm $\mathcal{A}$

$$\mathrm{P}\left[\mathrm{RSA}_{f,\mathcal{A}}(n)\right] \simeq 0$$

RSA assumption: There is RSA key generator for which RSA problem is hard

Notes

- Choice of e seems irrelevant (commonly $e = 3$ or $e = 2^{16} + 1$)
- How do we compute $\phi(\ell)$ fast? (for RSA key generator)
- How do we uniformly select $c \in \mathbb{Z}_\ell^*$? (for RSA game)

§10.4 Definition (Product group)

Let $\mathbb{G} = (G, \cdot, e)$ and $\mathbb{G}' = (G', \odot, e')$ be commutative groups. Then

$$\mathbb{G} \times \mathbb{G}' = \Big(G \times G', \otimes, (e, e')\Big)$$

with $(g_1, g_1') \otimes (g_2, g_2') = (g_1 \cdot g_2, g_1' \odot g_2')$ for all $g_1, g_2 \in G$ and $g_1', g_2' \in G'$

§10.5 Lemma (Product group)

$\mathbb{G} \times \mathbb{G}'$ is commutative group for all commutative groups $\mathbb{G}$ and $\mathbb{G}'$

Proof

Simple exercise $\square$

§10.6 Theorem (Chinese remainder theorem)

Let $p, p' \in \mathbb{N}$ be different primes and $\ell = pp'$.
Then $\mathbb{Z}_\ell^*$ and $\mathbb{Z}_p^* \times \mathbb{Z}_{p'}^*$ are isomorphic via

$$h\colon \mathbb{Z}_\ell^* \to \mathbb{Z}_p^* \times \mathbb{Z}_{p'}^* \quad \text{with} \quad h(a) = (a \bmod p,\ a \bmod p') \quad \text{for all } a \in \mathbb{Z}_\ell^*$$

Proof (1/2)

Let $a \in \mathbb{Z}_\ell^*$, so $\gcd(a, \ell) = 1$. Suppose that $a \bmod p \notin \mathbb{Z}_p^*$, so $\gcd(a, p) \neq 1$.
Then $\gcd(a, pp') = \gcd(a, \ell) \neq 1$, which is contradiction (analogous for p').
Obviously $h(1) = (1, 1)$. Let $a, b \in \mathbb{Z}_\ell^*$.

$$\begin{aligned} h(a \cdot_\ell b) &= (a \cdot_\ell b \bmod p,\ a \cdot_\ell b \bmod p') = (ab \bmod p,\ ab \bmod p') \\ &= \Big((a \bmod p) \cdot_p (b \bmod p),\ (a \bmod p') \cdot_{p'} (b \bmod p')\Big) \\ &= h(a) \otimes h(b) \end{aligned}$$

Proof (2/2)

By Lemma §9.9 there exist $q, q' \in \mathbb{Z}$ such that $pq + p'q' = 1$. So

$$pq = (1 - p'q') = 1 \bmod p' \qquad \text{and} \qquad p'q' = (1 - pq) = 1 \bmod p$$

Let $a \in \mathbb{Z}_p^*$ and $a' \in \mathbb{Z}_{p'}^*$. Then $\gcd(ap'q' + a'pq, \ell) = 1$ and

$$\begin{aligned} h(ap'q' + a'pq \bmod \ell) &= \left(ap'q' + a'pq \bmod p,\ ap'q' + a'pq \bmod p'\right) \\ &= \left(ap'q' \bmod p,\ a'pq \bmod p'\right) = (a, a') \end{aligned}$$

which proves that h is surjective. Finally, for injectivity, let $a, a' \in \mathbb{Z}_\ell^*$ be such that $h(a) = h(a')$. Hence $a = a' \bmod p$ and $a = a' \bmod p'$, so $p|(a - a')$ and $p'|(a - a')$. Since p and p' are prime, we obtain $pp'|(a - a')$, so $a = a' \bmod \ell$, which proves injectivity and thus h is isomorphism. □

Foundations

Let p, p' be different primes

Notes

- We can compute in $\mathbb{Z}_p^* \times \mathbb{Z}_{p'}^*$ instead of $\mathbb{Z}_{pp'}^*$
- $\mathbb{Z}_p^*$ and $\mathbb{Z}_{p'}^*$ are multiplicative groups of $\mathrm{GF}(p)$ and $\mathrm{GF}(p')$ respectively (trivial to represent since $\mathbb{Z}_p^* = \{1, \ldots, p-1\}$)
- $\phi(pp') = (p-1)(p'-1)$ by Theorem §10.3
- Simple to compute $\phi(\ell)$ given factorization $\ell = pp'$
- Simple to uniformly select $c \in \mathbb{Z}_{pp'}^*$
  by uniformly selecting $c_1 \in \mathbb{Z}_p^*$ and $c_2 \in \mathbb{Z}_{p'}^*$ and computing $h^{-1}(c_1, c_2)$

Foundations

Let p, p' be different primes

Decryption

- Should be easy with additional knowledge (factorization of $\ell = pp'$)
- Simple to compute $\sqrt[e]{c}$ given p and p' (or d or $\phi(pp')$)
  1. Compute order of $\mathbb{Z}^*_{pp'}$ which is $\phi(pp') = (p-1)(p'-1)$
  2. Compute inverse d of e in $\mathbb{Z}^*_{\phi(pp')}$ using extended Euclidian algorithm
  3. $\sqrt[e]{c} = c^d$ in $\mathbb{Z}^*_{pp'}$ because $(c^d)^e = c^{de} = c^{de \bmod \phi(pp')} = c^1 = c$ by Euler's theorem

- Important: We compute in $\mathbb{Z}^*_{pp'}$, but exponents calculated in $\mathbb{Z}^*_{\phi(pp')}$

Example

- Let $\ell = 4,187$, so we compute in $\mathbb{Z}_\ell^*$
- Let $m \in \mathbb{Z}_\ell^*$ be message, exponent $e = 5$, and $m^e = 133$
- Given factorization $\ell = 53 \cdot 79$, compute $\phi(\ell) = 52 \cdot 78 = 4,056$
- In $\mathbb{Z}_{4,056}^*$ compute $d = 5^{-1} = -811 = 3,245$

$$\gcd'(4,056, 5) = (d, y, x - 811y) \qquad (d, x, y) = \gcd'(5, 1) = (1, 0, 1)$$

so $1 = 1 \cdot 4,056 - 811 \cdot 5$

- Compute message by $m = 133^d$

$$\begin{aligned} 133^{3,245} &= 133^{2048} \cdot 133^{1,024} \cdot 133^{128} \cdot 133^{32} \cdot 133^{8} \cdot 133^{4} \cdot 133^{1} \\ &= 2,533 \cdot 2,943 \cdot 579 \cdot 2,144 \cdot 1,690 \cdot 2,204 \cdot 133 = 127 \end{aligned}$$

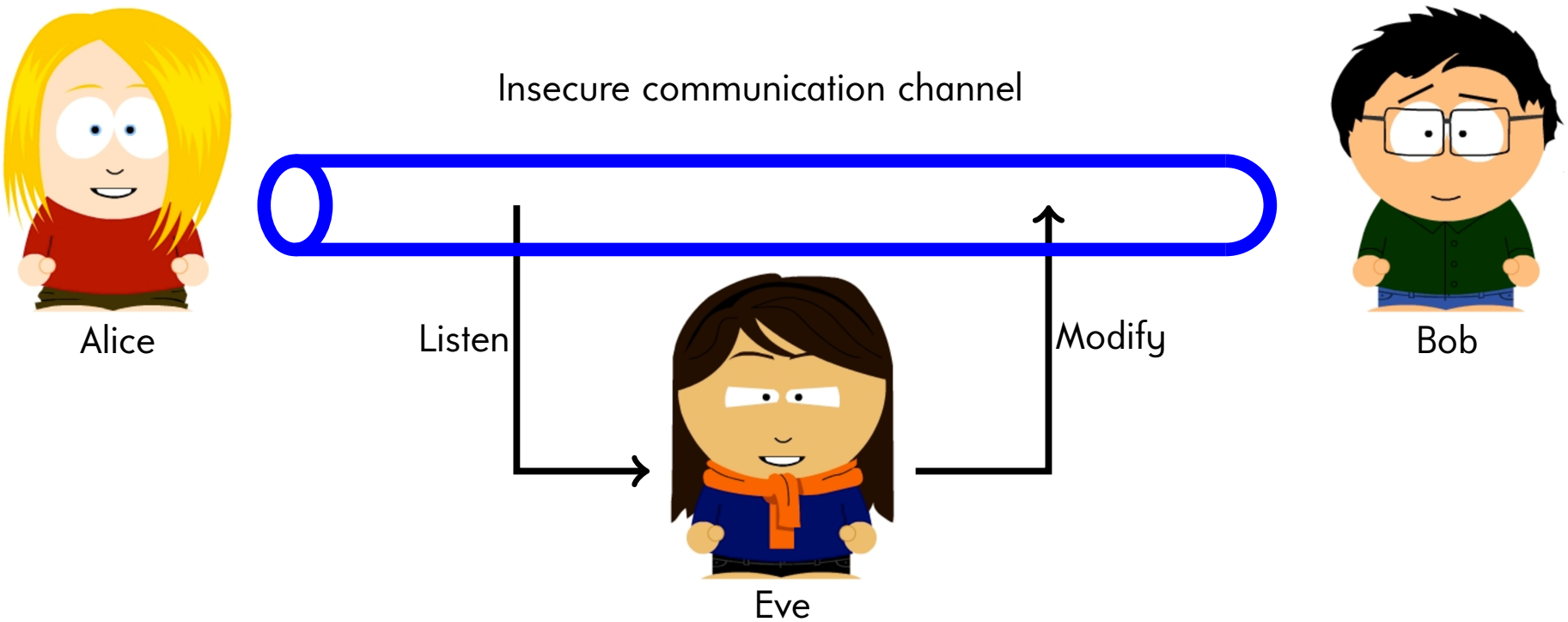
RSA Assumption

Summary

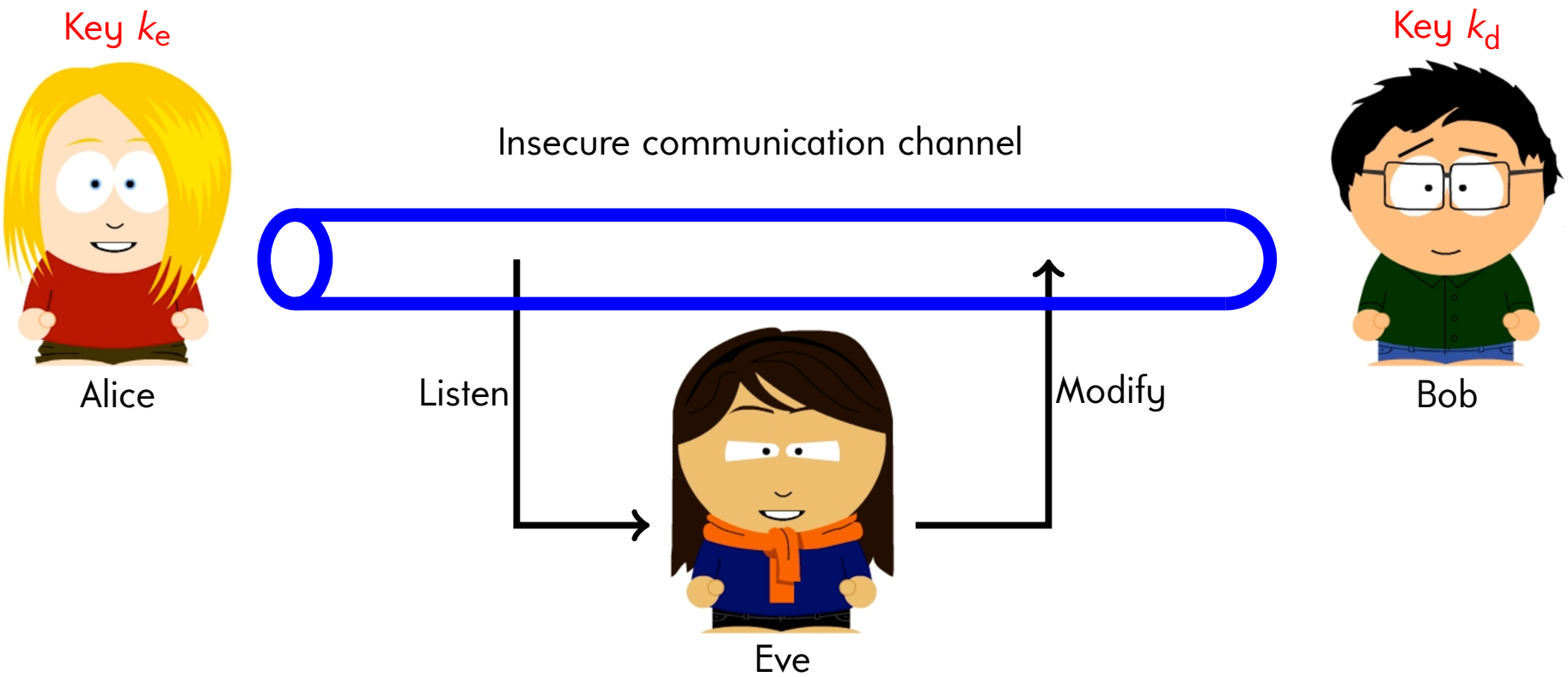
- RSA assumption requires factoring to be hard
  (if factoring is simple, then adversary can factor ℓ into p and p')
- RSA problem might be simpler than factoring
  (maybe there is better way to compute $\sqrt[e]{c}$ without factoring ℓ)
- Currently RSA problem & factoring are assumed to be "equally" hard
  (which does not say that they are hard)

Public-Key Encryption

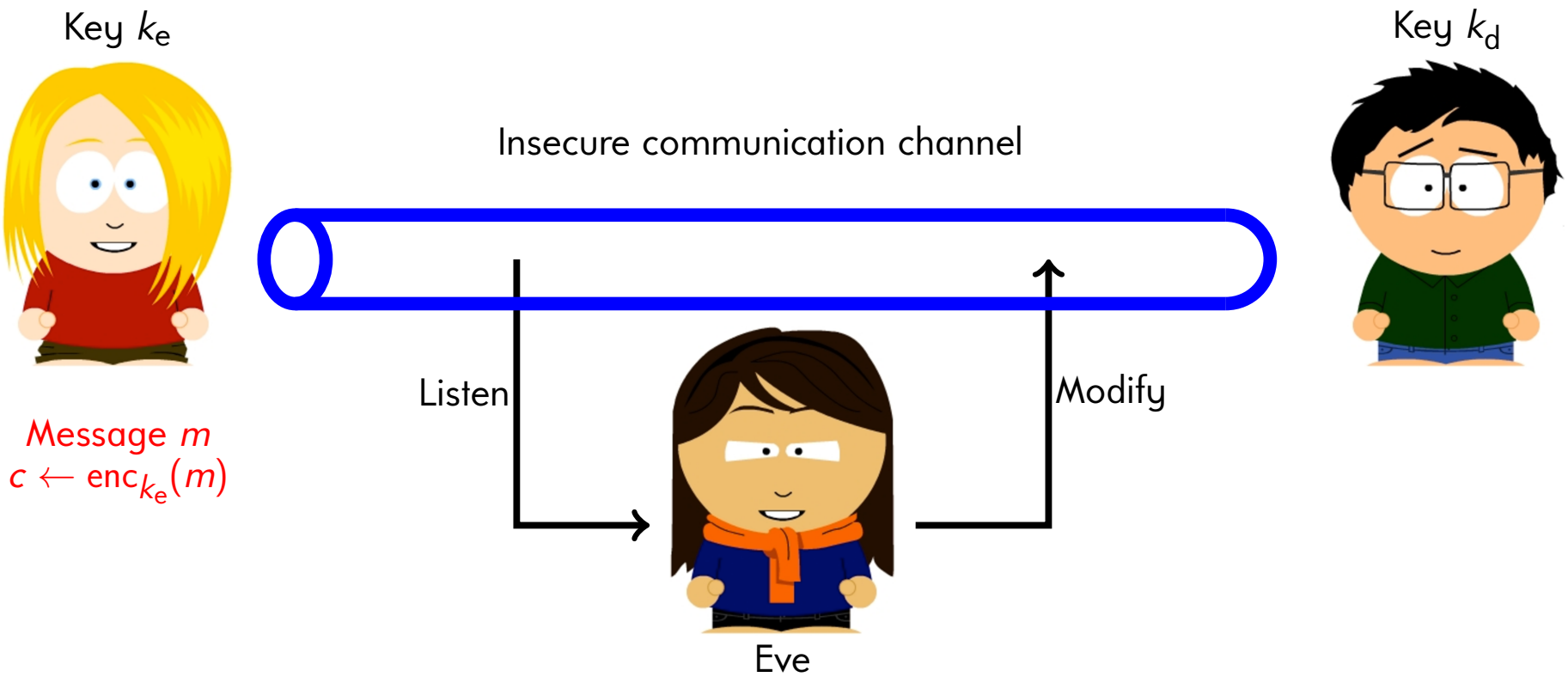
$\langle k_e, k_d \rangle \leftarrow \text{gen}(1^n)$



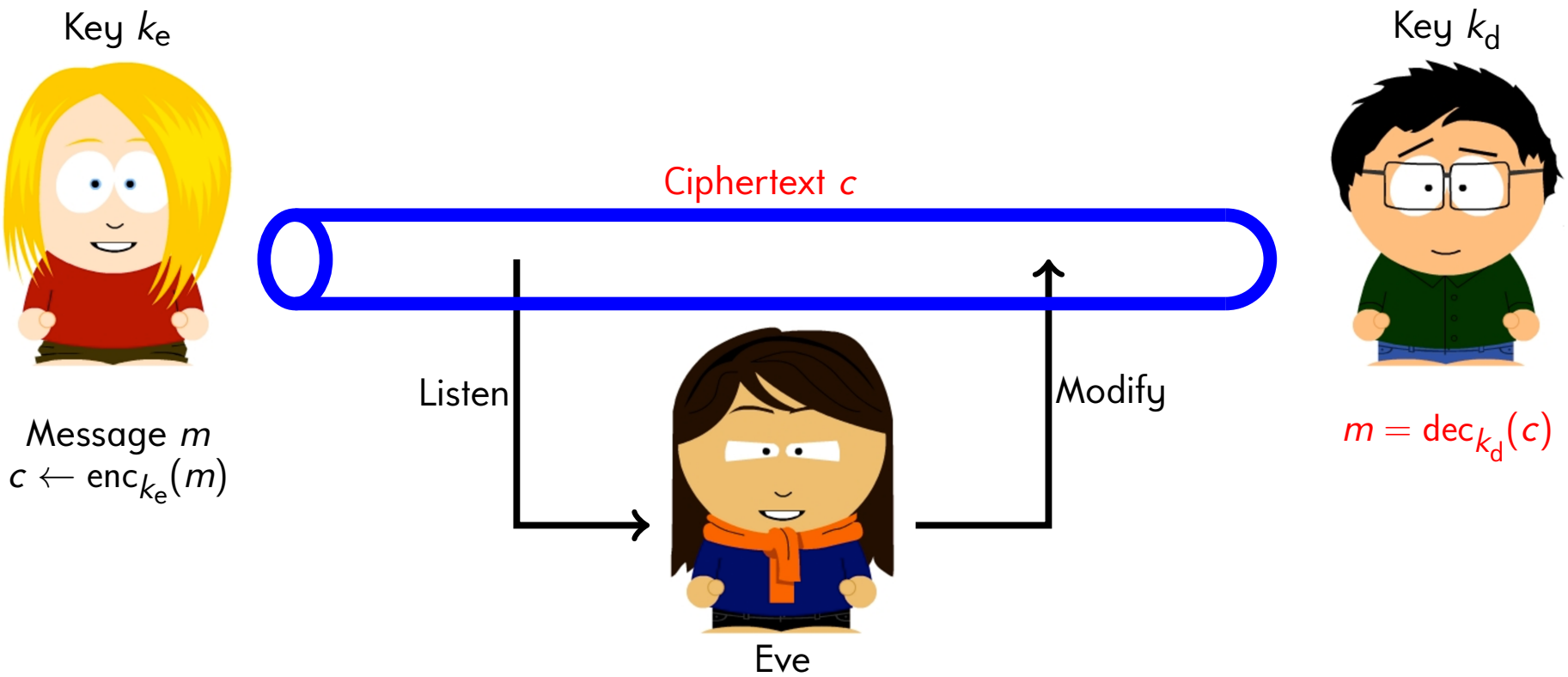
Public-Key Encryption



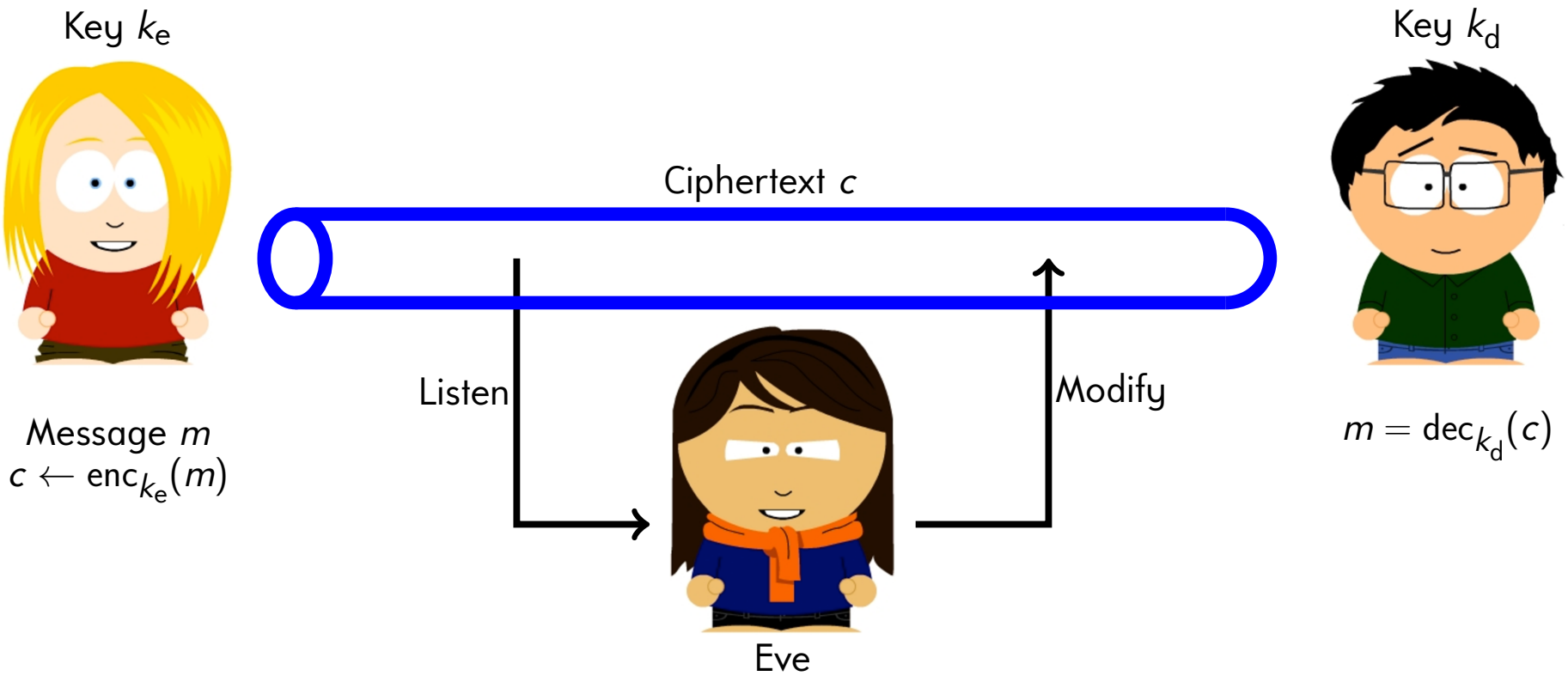
Public-Key Encryption



Public-Key Encryption



Public-Key Encryption



Notes

- 2-part key (encryption key = public key; decryption key = secret key)
- Key distribution later

Public-Key Encryption

Notes

- Public (encryption) key can be shared with anyone
  (traditionally written pk)
- Secret (decryption) key only known to generating entity
  (traditionally written sk)
- Asymmetric nature of sub-keys $\rightarrow$ asymmetric cryptography

Public-Key Encryption

§10.7 Definition (cf. §3.6; Efficient public-key encryption scheme)

(Efficient) public-key encryption scheme is triple $(\mathsf{gen}, \mathsf{enc}, \mathsf{dec})$ of probabilistic polynomial-time computable functions

- $\mathsf{gen} \colon R \times \{1\}^* \to K^2$ i.e. $(\mathrm{P}^n_{K^2})_{n \in \mathbb{N}}$ of distributions $\mathrm{P}^n_{K^2} \colon K^2 \to [0,1]$
- $\mathsf{enc} \colon R \times K \times M \to C$ and $\mathsf{dec} \colon K \times C \to (M \cup \{\bot\})$
- $|k_\mathsf{e}| \geq n$ and $|k_\mathsf{d}| \geq n$ for all $r \in R$, $n \in \mathbb{N}$, and $\langle k_\mathsf{e}, k_\mathsf{d} \rangle = \mathsf{gen}(r, 1^n)$
- $\mathsf{dec}_{k_\mathsf{d}}(c) = m$ for all $\langle k_\mathsf{e}, k_\mathsf{d} \rangle \in \mathrm{ran}(\mathsf{gen})$, $m \in M$, and $c \in C$ with $\mathrm{P}\big[c \leftarrow \mathsf{enc}_{k_\mathsf{e}}(m)\big] \neq 0$ (perfect correctness)

Notes

- Key generation generates public and secret key
- Generated keys each have length at least n
- Encryption with public key k_e and decryption with secret key k_d

Public-Key Encryption

Alice

$\langle k_e, k_d \rangle \leftarrow \text{gen}(1^n)$

Eve

Public-Key Encryption

Alice

$\langle k_e, k_d \rangle \leftarrow \text{gen}(1^n)$

Public key k_e

Eve

Public-Key Encryption

Alice

$\langle k_e, k_d \rangle \leftarrow \text{gen}(1^n)$

Public key k_e

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

Eve

Public-Key Encryption

Alice

$\langle k_e, k_d \rangle \leftarrow \text{gen}(1^n)$

$b \leftarrow U_1$

Public key k_e

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

Eve

Public-Key Encryption

Alice

$\langle k_e, k_d \rangle \leftarrow \text{gen}(1^n)$

$b \leftarrow U_1$

$c \leftarrow \text{enc}_{k_e}(m_b)$

Public key k_e

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

Eve

Public-Key Encryption

Alice

$\langle k_e, k_d \rangle \leftarrow \text{gen}(1^n)$

$b \leftarrow U_1$

$c \leftarrow \text{enc}_{k_e}(m_b)$

Public key k_e

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

Ciphertext c

Eve

Public-Key Encryption

Alice

$\langle k_e, k_d \rangle \leftarrow \text{gen}(1^n)$

$b \leftarrow U_1$

$c \leftarrow \text{enc}_{k_e}(m_b)$

Public key k_e

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

Ciphertext c

Eve

Determines b'

Public-Key Encryption

Alice

$\langle k_e, k_d \rangle \leftarrow \text{gen}(1^n)$

$b \leftarrow U_1$

$c \leftarrow \text{enc}_{k_e}(m_b)$

Public key k_e

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

Ciphertext c

Bit b'

Eve

Determines b'

Public-Key Encryption

Alice

$\langle k_e, k_d \rangle \leftarrow \text{gen}(1^n)$

$b \leftarrow U_1$

$c \leftarrow \text{enc}_{k_e}(m_b)$

Return $b \stackrel{?}{=} b'$

Public key k_e

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

Ciphertext c

Bit b'

Eve

Determines b'

Public-Key Encryption

§10.8 Definition (Indistinguishability game)

Let $n \in \mathbb{N}$, $\mathcal{E} = (\mathrm{gen}, \mathrm{enc}, \mathrm{dec})$ efficient public-key encryption scheme and $\mathcal{A}$ stateful algorithm. Adversarial indistinguishability game $\mathrm{PubK}^{\mathrm{one}}_{\mathcal{E},\mathcal{A}}(n)$ is

1. $\langle k_e, k_d \rangle \leftarrow \mathrm{gen}(1^n)$; public key k_e sent to adversary $\mathcal{A}$
2. $\mathcal{A}$ selects messages $m_0, m_1 \in M$ with $|m_0| = |m_1|$
3. $b \leftarrow U_1$; challenge ciphertext $c \leftarrow \mathrm{enc}_{k_e}(m_b)$ sent to $\mathcal{A}$
4. $\mathcal{A}$ outputs bit $b' \in \{0,1\}$
5. Return 1 if $b = b'$ ($\mathcal{A}$ wins) and return 0 otherwise

Changes

- $\mathcal{A}$ receives public key k_e & selects messages of equal length
- Security parameter n not sent to $\mathcal{A}$; public key k_e longer than n

§10.9 Definition (CPA-secure)

Public-key encryption scheme $\mathcal{E} = (\mathsf{gen}, \mathsf{enc}, \mathsf{dec})$ is CPA-secure if for every PPT algorithm $\mathcal{A}$ (in public key length $n = |k_e|$)

$$\mathsf{P}\left[\mathsf{PubK}^{\mathsf{one}}_{\mathcal{E},\mathcal{A}}(n)\right] \simeq \frac{1}{2}$$

Notes

- No access to encryption oracle
  (unnecessary since $\mathcal{A}$ can encrypt with known public key)
- Perfectly secret schemes impossible
- Deterministic schemes again cannot be CPA-secure
- CPA-security again implies CPA-security for multiple messages

Public-Key Encryption

§10.10 Definition (CCA-secure)

Public-key encryption scheme $\mathcal{E} = (\mathsf{gen}, \mathsf{enc}, \mathsf{dec})$ is CCA-secure
if for every PPT algorithm $\mathcal{A}$ with access to decryption oracle

$$\mathrm{P}\left[\mathrm{PubK}^{\mathrm{one}}_{\mathcal{E},\mathcal{A}}(n)\right] \simeq \frac{1}{2}$$

where $\mathcal{A}$ must not query decryption oracle on challenge ciphertext
after it is received by $\mathcal{A}$

Note

- Also known as IND-CCA2 (IND = indistinguishability)
  (IND-CCA1 = no decryption oracle after receipt of challenge ciphertext)

RSA Cipher

§10.11 Definition (Plain RSA)

Let f be RSA key generator, for which RSA problem is hard.
Public-key encryption scheme $\mathsf{RSA} = (\mathsf{gen}, \mathsf{enc}, \mathsf{dec})$ given by

- $\mathsf{gen}(1^n)$ runs $(\ell, e, d) \leftarrow f(1^n)$ and returns $\langle (\ell, e), (\ell, d) \rangle$
- $\mathsf{enc}_{\langle \ell, e \rangle}(m) = m^e \bmod \ell$ for all $m \in \mathbb{Z}_\ell^*$
- $\mathsf{dec}_{\langle \ell, d \rangle}(c) = c^d \bmod \ell$ for all $c \in \mathbb{Z}_\ell^*$

Is RSA CPA-secure?

✗ As deterministic cipher it cannot be CPA-secure
(also not EAV-secure as EAV-secure = CPA-secure for asymmetric ciphers)

Summary

- RSA assumption
- RSA key generation
- Plain RSA cipher