NUMERI PRIMI E CRITTOGRAFIA

Parte I. Crittografia a chiave simmetrica

dall'antichità all'era del computer

Parte II. Note della Teoria dei Numeri

concetti ed algoritmi a supporto della Crittografia

Parte III. Crittografia a chiave pubblica

il superamento del problema dello scambio delle chiavi

Parte IV. Esercitazione di gruppo

implementazione di un minisistema crittografico RSA

Tutor: Franco Danielli (franco.danielli@tin.it)

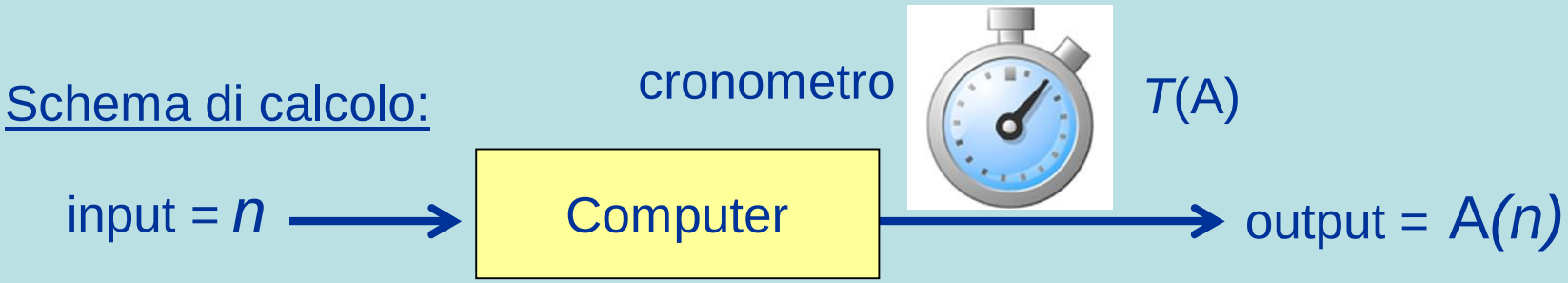
PARTE III

SISTEMI CRITTOGRAFICI A CHIAVE PUBBLICA

- Concetto di Complessità Computazionale
- L'idea di Diffie-Hellman: funzione unidirezionale con il *trapdoor*
- Funzioni unidirezionali in Aritmetica Modulare
- Il Metodo RSA
- Strategie d'attacco al sistema RSA, sicurezza
- Sicurezza crittografica e potenza computazionale
- La firma digitale: autenticità e completezza del messaggio
- Stato dell'arte della Crittografia moderna

Concetto di Complessità Computazionale di un algoritmo

Schema di calcolo:



dove
- n = argomento (esempio: $n \approx 5.000.000.000$)
- L(n) = numero di cifre di n (esempio: L(n) = 10)
- A(n) = algoritmo di calcolo di argomento n
- T(A) = tempo impiegato dal computer ad eseguire il calcolo

- Se T(A) = f (L(n)) → allora l'algoritmo A(n) si dice efficiente, ovvero a tempo polinomiale
- Se T(A) = f (n) → allora l'algoritmo A(n) si dice inefficiente, ovvero a tempo esponenziale
- Se T(A) ≈ mesi/anni si dice anche che il problema A(n) è intrattabile

Esempio: Complessità Computazionali a confronto

Algoritmo A: Moltiplicazione
input: n_1, n_2 output: n3 = n1 · n2

$$T(A) = k \cdot L(n_1) \cdot L(n_2)$$

n_1, n_2	10^{10}	10^{20}	10^{30}	10^{40}
$L(n_1)\cdot L(n_2)$	100	400	900	1.600
T(A)	1 ms	4 ms	9 ms	16 ms

Alg. B: Fattorizzazione (per tentativi)
input: n_3; output: n_1, n_2 con $n_1 \cdot n_2 = n_3$

$$T(B) = k \cdot \sqrt{n_3}$$

n_3	10^{10}	10^{20}	10^{30}	10^{40}
$\sqrt{n_3}$	10^5	10^{10}	10^{15}	10^{20}
T(B)	1 ms	1,7 min	116 d	>30.000 y

- L'algoritmo A (Moltiplicazione) è efficiente (a tempo polinomiale)
- L'algoritmo B (Fattorizzazione) è inefficiente (a tempo esponenziale)
- La Fattorizzazione diventa rapidamente un problema intrattabile

L'idea di Diffie-Hellman: Funzione unidirezionale con il "*Trapdoor*"

Esempio concettuale: manipolazione di un lucchetto

Concetto di Crittografia a Chiave Pubblica (Diffie-Hellman, 1976)

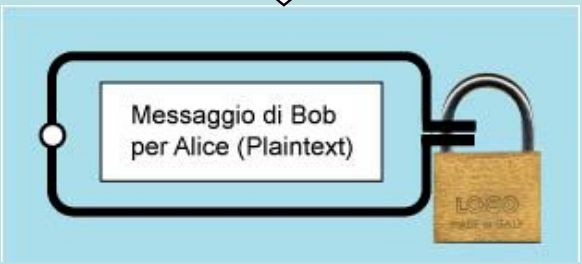
ALICE

Alice possiede l'unica chiave (chiave privata) che apre tutti i suoi lucchetti (il *trapdoor*)

Alice apre la scatola con la sua chiave privata e recupera il *plaintext*

BOB

- Crea il messaggio per Alice
- Lo mette in una scatola
- Si fa dare un lucchetto di Alice
- Chiude il lucchetto
- Spedisce la scatola



Lucchetti di Alice (tutti identici) a disposizione dei mittenti:

Implementazione del Sistema crittografico a chiave pubblica

Diffie ed Hellman presentarono l'idea del sistema a chiave pubblica, ma non riuscirono ad esibire una funzione unidirezionale (con il *trapdoor*) idonea a realizzarlo.

Ci riuscirono invece nel 1978 altri tre ricercatori americani del MIT di Boston: Rivest, Shamir, Adleman.

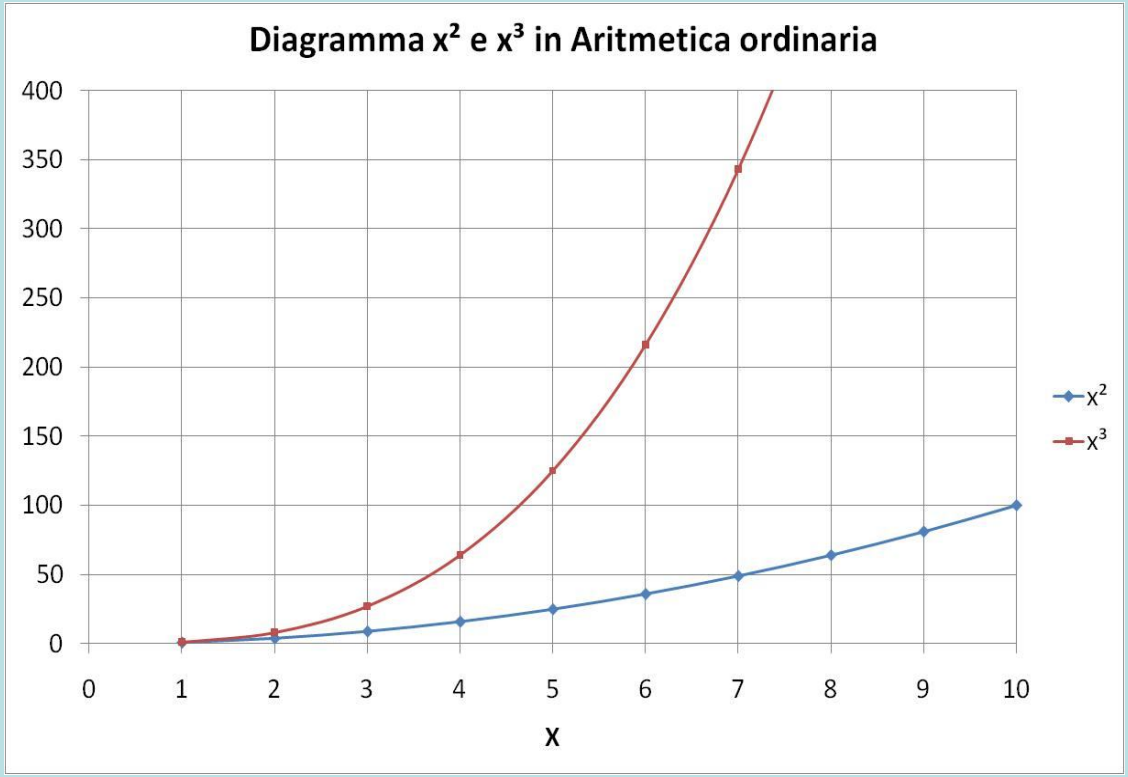
Nacque così il Sistema RSA, dalle iniziali dei suoi creatori.

A tutt'oggi, RSA è il sistema crittografico a chiave pubblica più diffuso ed affermato nel mondo.

Ricerca di funzioni unidirezionali per applicazioni in Crittografia (1)

Funzioni f(x) = x^n in Aritmetica ordinaria

x	x^2	x^3
1	1	1
2	4	8
3	9	27
4	16	64
5	25	125
6	36	216
7	49	343
8	64	512
9	81	729
10	100	1000

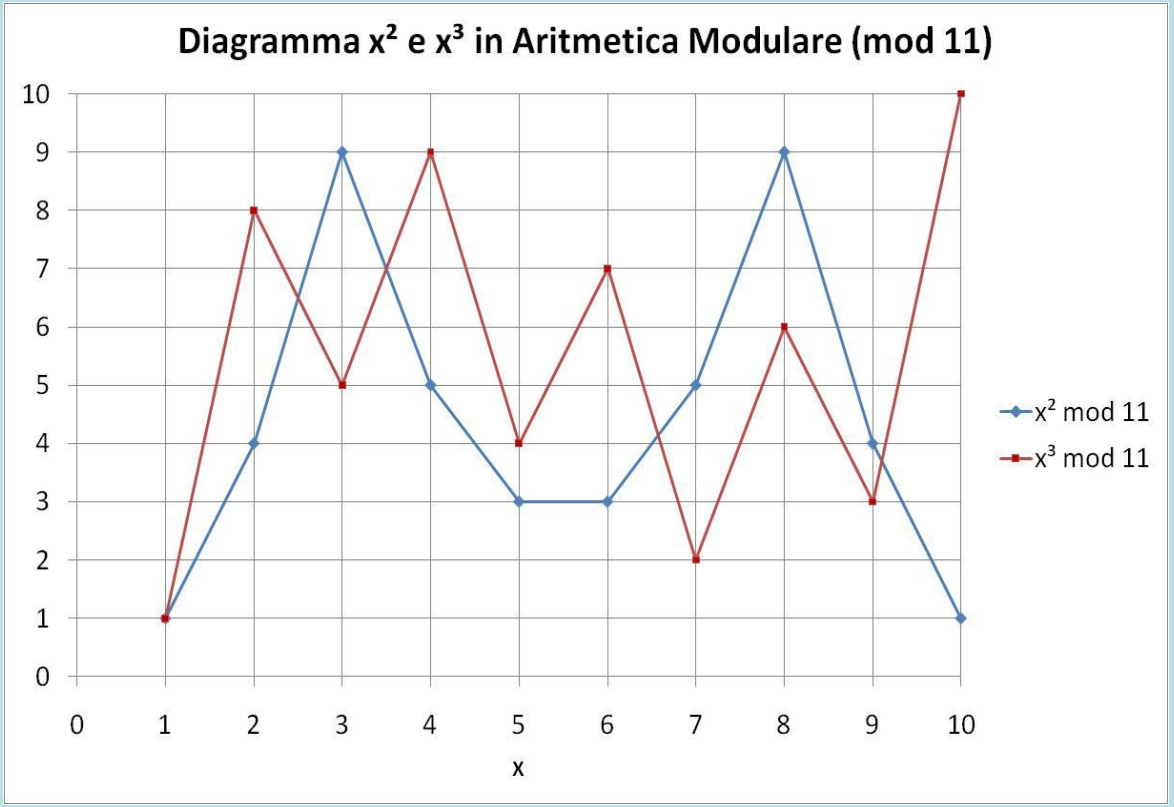


In Aritmetica ordinaria, funzioni del tipo f(x)= x^n sono regolari, monotone e facilmente invertibili: non possono essere funzioni unidirezionali

Ricerca di funzioni unidirezionali per applicazioni in Crittografia (2)

Funzioni f(x) = x^n in Aritmetica Modulare

x	x^2 mod 11	x^3 mod 11
1	1	1
2	4	8
3	9	5
4	5	9
5	3	4
6	3	7
7	5	2
8	9	6
9	4	3
10	1	10



In Aritmetica Modulare, funzioni del tipo f(x)= x^n sono irregolari, caotiche ed imprevedibili:
ottime candidate ad essere funzioni unidirezionali

Sistema RSA (Rivest, Shamir, Adleman - USA 1978)

A. Ogni utente del sistema crittografico costruisce le proprie chiavi, pubblica e privata

	Creazioni delle chiavi di Alice	Miniesempio
1	Sceglie due grandi numeri primi p, q	$p = 31$, $q = 53$
2	Calcola il prodotto $n = p \cdot q$	$n = 31 \times 53 = 1643$
3	Calcola la funzione di Eulero $\Phi(n)$	$\Phi(n) = 30 \times 52 = 1560$
4	Sceglie un numero e coprimo con $\Phi(n)$	$e = 439$
5	Calcola d, inverso di e modulo $\Phi(n)$	$d = 199$
6	Rende pubblica la coppia (n, e)	**1643, 439** Chiave pubblica
7	Tiene segreto il dato d	**199** Chiave segreta
8	Distrugge tutti i dati intermedi p, q, $\Phi(n)$	~~31, 53, 1560~~

Sistema RSA (Rivest, Shamir, Adleman - USA 1978)

B. Cifratura di un messaggio nel Sistema RSA

	Bob vuole trasmettere un messaggio riservato ad Alice	Miniesempio
1	Scrive il *plaintext* da trasmettere	*plaintext* = " w "
2	Si procura la chiave pubblica autentica di Alice	$n = 1643$, $e = 439$
3	Trasforma il *plaintext* in un numero m con il codice ASCII, verifica che $m < n$	m = ASC(w) = 119 < 1643
4	Crea il crittogramma c col calcolo della potenza modulare $c = m^e \bmod n$	$c = 119^{439} \bmod 1643 = 1049$
5	Spedisce il crittogramma c ad Alice sul canale insicuro	Bob $c = 1049$ → Alice

Sistema RSA (Rivest, Shamir, Adleman - USA 1978)

C. Decifrazione di un crittogramma nel Sistema RSA

	Alice decifra il crittogramma ricevuto da Bob con la propria chiave privata	Miniesempio
1	Alice ha ricevuto un crittogramma *c* da Bob	$c = 1049$
2	Per decifrare, Alice fa uso della propria chiave segreta *d* (oltre al modulo *n*)	$n = 1643$, $d = 199$
3	Decifra il crittogramma col calcolo della potenza modulare $m = c^d \bmod n$	$m = 1049^{199} \bmod 1643 = 119$
4	Recupera il *plaintext* decodificando *m* con il codice ASCII	*plaintext* = Character(119) = “ *w* “

Il sistema RSA funziona sulla base del Teorema di Eulero:

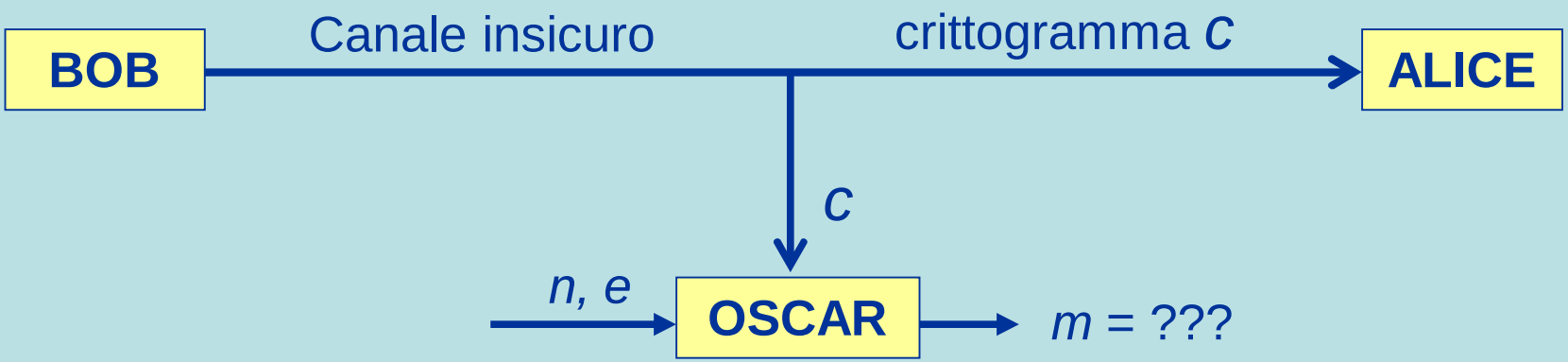
Teorema di Eulero: $a^{1+\Phi(n)} \equiv a \pmod{n}$, a intero, n modulo > 0

Cifratura e decifrazione nel Sistema RSA:

messaggio:	m
cifratura di m:	$c = m^e \bmod n$
decifrazione di c:	$x = c^d \bmod n = m^{e \cdot d} \bmod n$
ma d è l'inverso di e mod $\Phi(n)$:	$e \cdot d \equiv 1 \pmod{\Phi(n)}$
ovvero:	$e \cdot d = 1 + k \cdot \Phi(n)$
da cui:	$x = m^{1+k \cdot \Phi(n)} \equiv m \pmod{n}$
e poiché $m < n$:	$x = m$

con ciò, il destinatario recupera il messaggio.

Attacco al Sistema RSA (*ciphertext only*)

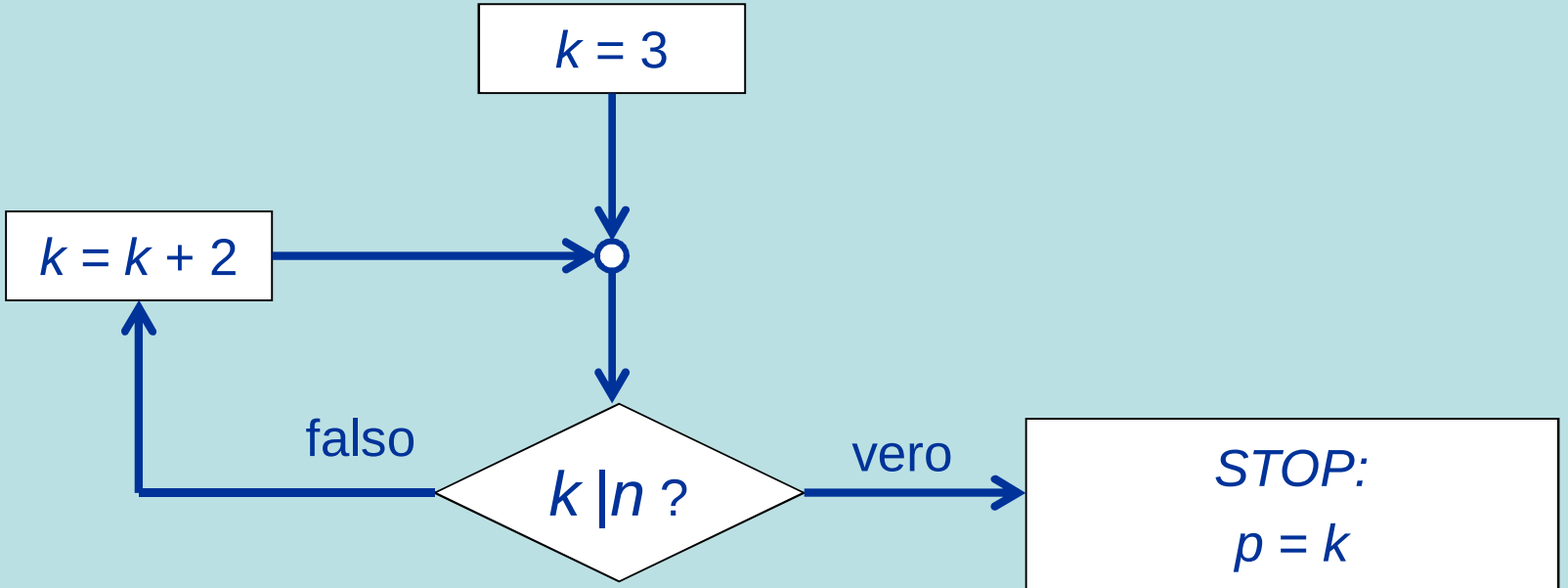


- Oscar ha intercettato sul canale insicuro il crittogramma c indirizzato ad Alice
- Oscar conosce inoltre, come tutti, la chiave pubblica di Alice (n, e)
- Oscar sa anche che: $c = m^e \bmod n$, $m = c^d \bmod n$
- ma Oscar ignora la chiave segreta d (il "*trapdoor*" di Alice)

Che cosa può fare Oscar per forzare il sistema crittografico ed impadronirsi del *plaintext* m ?

1° strategia di attacco: Fattorizzazione di *n*

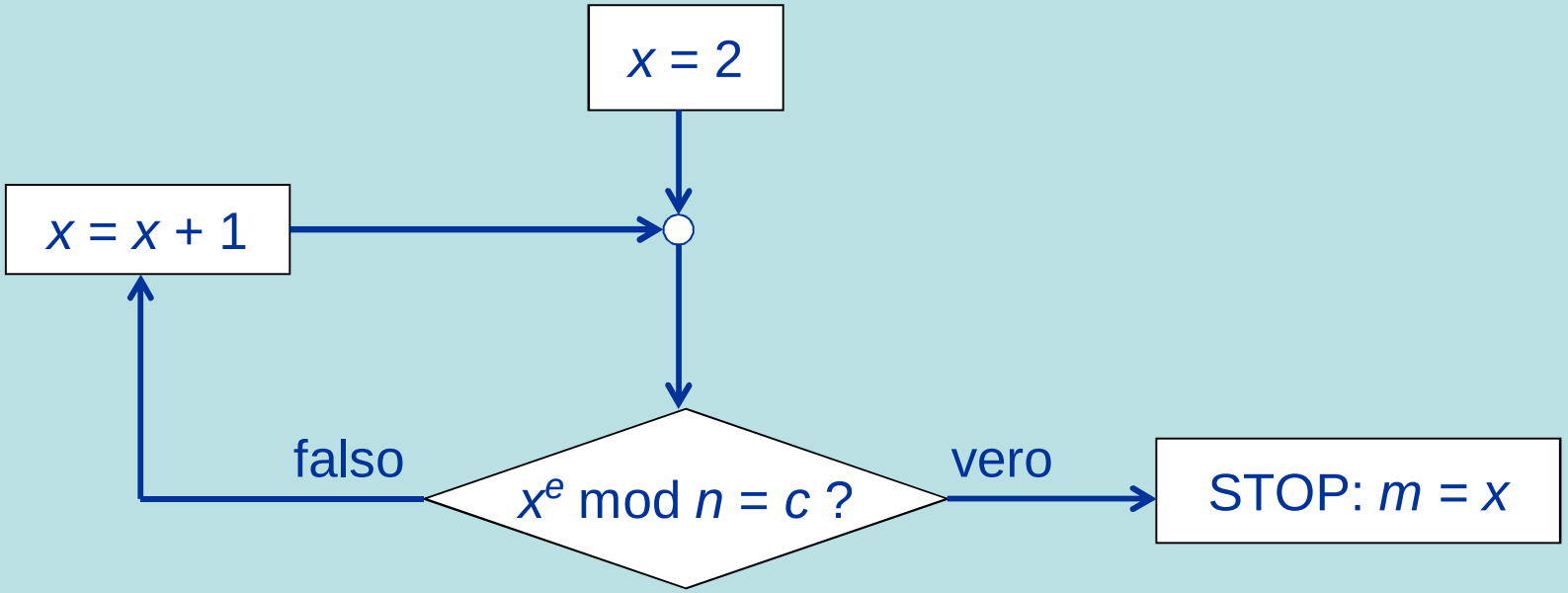
Oscar può provare a fattorizzare il modulo *n* per ricostruire la chiave privata *d* di Alice:



Tuttavia, per la fattorizzazione è noto solo l'algoritmo (inefficiente) per tentativi.

2° strategia di attacco: Calcolo di *m* per tentativi

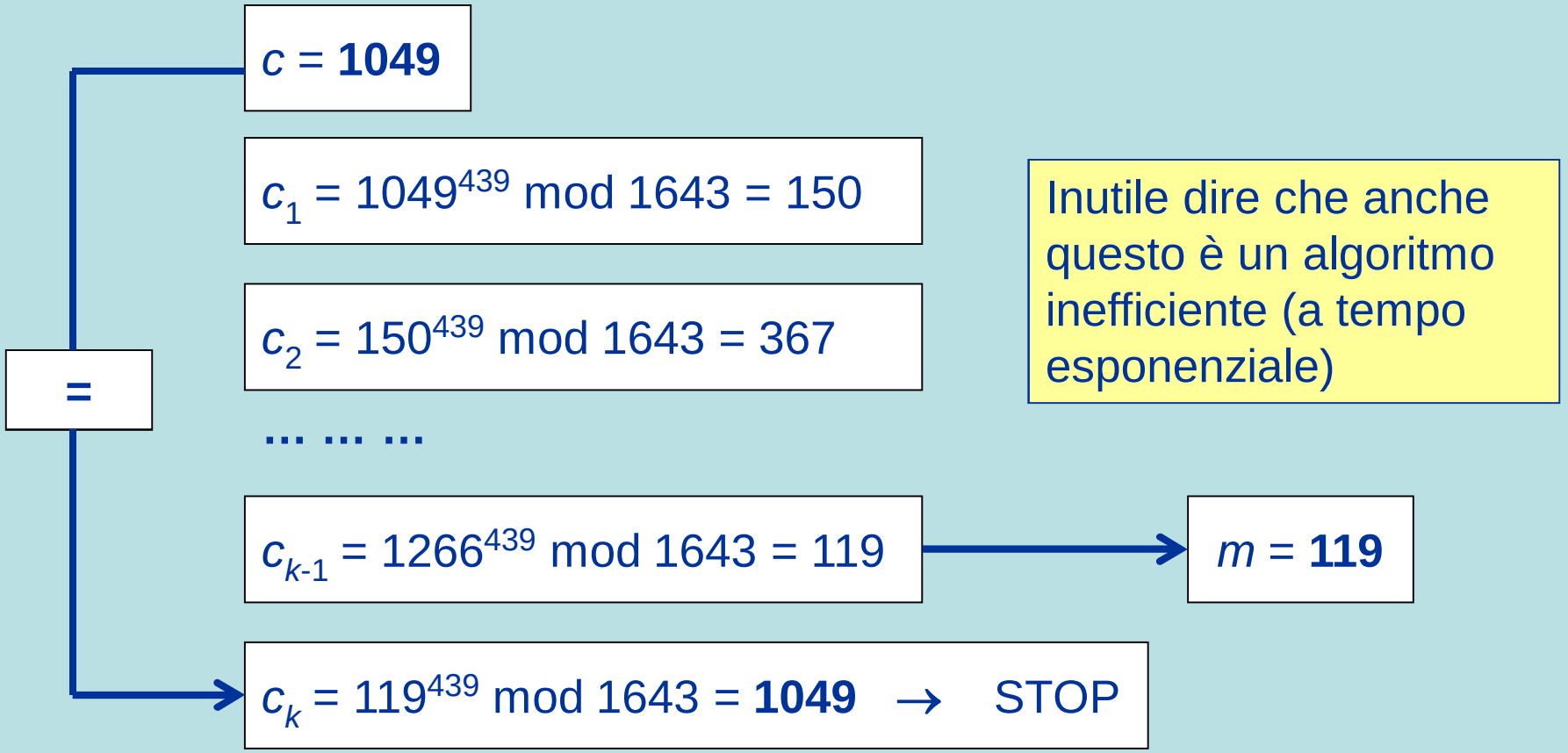
Oscar può procedere per tentativi a trovare quel numero *x* la cui cifratura con la chiave pubblica di Alice dà luogo al crittogramma *c*:



Questo è un algoritmo per tentativi (forza bruta), a tempo esponenziale (cioè, inefficiente)

3° strategia: Calcolo di *m* per attacco ciclico

Partendo da *c*, Oscar applica iterativamente l'algoritmo di cifratura fino a ritrovare *c* stesso. Allora, l'iterato precedente è precisamente il messaggio *m* cercato. Con riferimento al miniesempio (*n* = 1643, *e* = 439, *c* = 1049) :



Inutile dire che anche questo è un algoritmo inefficiente (a tempo esponenziale)

Sicurezza del sistema crittografico RSA

La sicurezza del sistema RSA nei confronti di un attacco si fonda sul fatto che, ad oggi, non si conoscono algoritmi efficienti per le attività di Crittoanalisi:

Algoritmi di Crittografia (costruzione delle chiavi, cifratura, decifrazione)
❑ Test di Primalità
❑ Algoritmo Euclideo
❑ Inverso modulo n
❑ Potenza modulo n

↓

Algoritmi efficienti
(a tempo polinomiale)

Algoritmi di Crittoanalisi (attacco al sistema)
❑ Fattorizzazione del modulo
❑ Ricerca del messaggio per tentativi
❑ Attacco ciclico

↓

Algoritmi inefficienti
(a tempo esponenziale)

Quale strategia resta ad Oscar?

- Il sistema RSA è (almeno attualmente) al sicuro da attacchi tipo *ciphertext only*.
- Ad Oscar resta tuttavia una possibilità: quella di spacciarsi per uno dei due partner (Alice o Bob), sostituendo una propria chiave pubblica a quella del destinatario.
- Per questo motivo, nel sistema RSA occorre che il mittente accerti sempre la chiave pubblica autentica del destinatario prima di cifrare il messaggio e spedire il crittogramma.

La firma digitale nel sistema crittografico

La firma autografa identifica il mittente di una lettera, o il contraente di un contratto.

La firma “digitale” ne ha tutte le caratteristiche:

- Autenticazione del mittente (accertamento della sua identità)
- Verifica dell’integrità del messaggio (accertamento che il messaggio non sia stato manipolato da altri)
- Non ripudiabilità del messaggio (accertamento della responsabilità del mittente)

Nel sistema RSA la firma digitale si può realizzare in modo molto semplice, scambiando il ruolo delle chiavi.

Firma digitale nel sistema RSA

	Procedura	Miniesempio		
1	Alice costruisce le sue chiavi RSA	$n_A = 1643$	$e_A = 439$	$d_A = 199$
2	Bob costruisce le sue chiavi RSA	$n_B = 1769$	$e_B = 247$	$d_B = 823$
3	Bob crea il messaggio $m_B = 119$ per Alice e ne calcola il crittogramma $c_B = m_B^{eA} \bmod n_A$	**$m_B = 119$** **$c_B = 119^{439} \bmod 1643 = 1049$**		
4	Bob firma il crittogramma c_B con la sua chiave privata $s_B = c_B^{dB} \bmod n_B$	**$c_B = 1049$** **$s_B = 1049^{823} \bmod 1769 = 1550$**		
5	Alice riceve da Bob il crittogramma firmato (s_B, c_B)	Bob **(1550, 1049)** → Alice		
6	Alice verifica la firma con la chiave pubblica di Bob: $s_B^{eB} \equiv c_B \pmod{n_B}$?	**$1550^{247} \equiv 1049 \pmod{1769}$?** **$\Rightarrow$ Vero !**		
7	Se si, Alice decifra il crittogramma con la propria chiave segreta: $m_B = c_B^{dA} \bmod n_A$	**$m_B = 1049^{199} \bmod 1643 = 119$**		
8	Se no, Alice rifiuta il messaggio:	il mittente non è Bob, oppure il messaggio è stato manipolato		

Sicurezza crittografica e Progresso tecnologico

Ipotesi di implementazione RSA su computer, oggi:

Frequenza CPU	Modulo RSA	Tempo Crittografia	Tempo Crittoanalisi	Sicurezza
GHz (10^9 op/s)	$n \approx 10^{100}$	$T = k \cdot L(n)$ = 1s	$T = k \cdot \sqrt{n}$ = 10.000h	**OK**

Progresso tecnologico dei computer, fra 10÷20 anni:

THz (10^{12} op/s)	$n \approx 10^{100}$	$T = \underline{k} \cdot L(n)$ = 1ms	$T = \underline{k} \cdot \sqrt{n}$ = 10h	KO !!

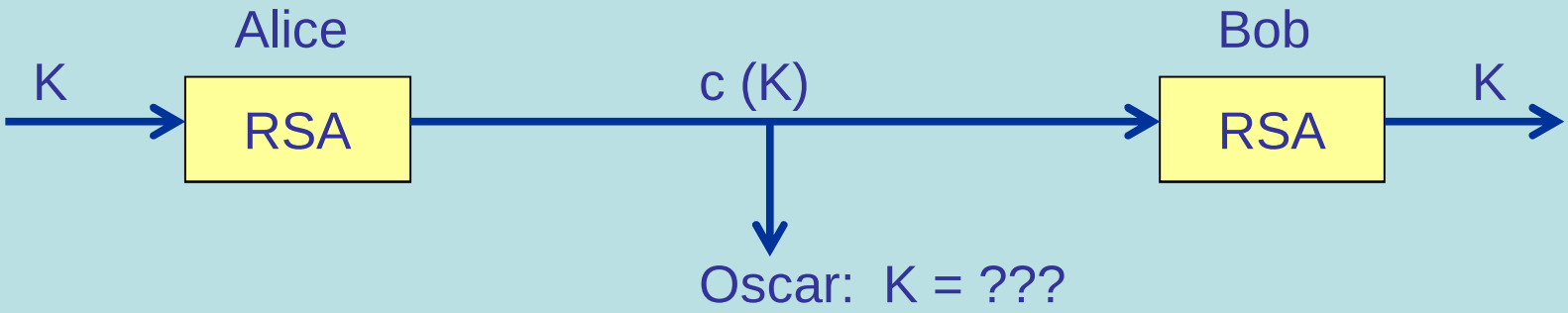
Ma basta aumentare di un po' le cifre del modulo n:

THz (10^{12} op/s)	$n \approx 10^{106}$	$T = \underline{k} \cdot L(n)$ = 1,06 ms	$T = \underline{k} \cdot \sqrt{n}$ = 10.000h	**OK**

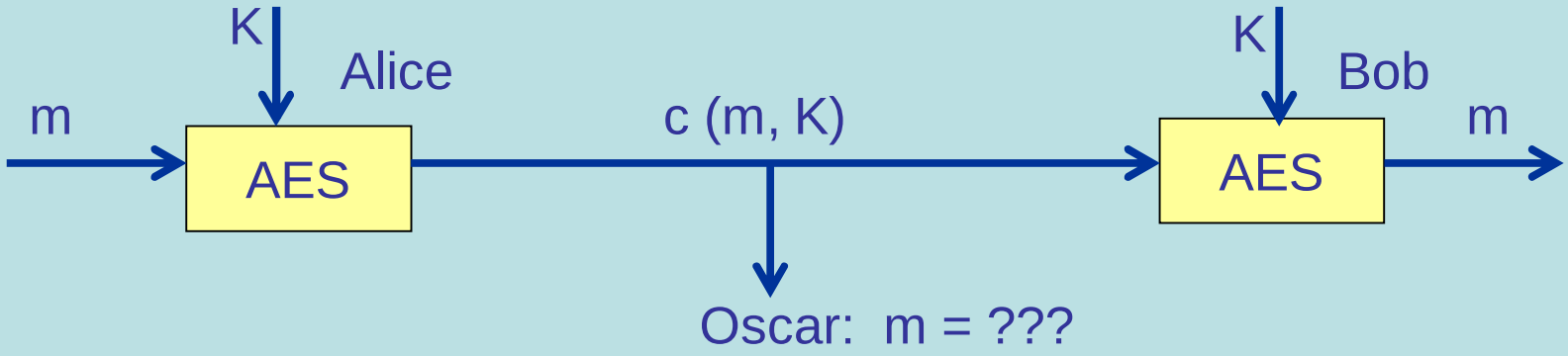
Conclusione: il progresso tecnologico (velocità di calcolo dei computer) favorisce più la Crittografia della Crittoanalisi.

Stato dell'arte della Crittografia moderna

Il sistema RSA a chiave pubblica costituisce la soluzione definitiva dello scambio delle chiavi fra Alice e Bob:



Dopo lo scambio, la comunicazione sul canale insicuro avviene con un sistema a chiave simmetrica, tipo DES, AES (più veloce):



Crittografia e Matematica

È un problema matematico mai risolto:

la fattorizzazione di un numero composto

ad aver dato realtà e successo all'intuizione geniale di Diffie ed Hellman:

il sistema crittografico a chiave pubblica