Codici Segreti: Un viaggio nella Crittografia

Agostino Dovier

Dip di Scienze Matematiche, Informatiche e Fisiche
CLP Lab
Univ. di Udine

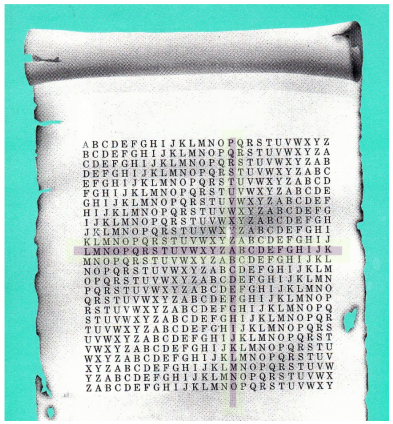
Aprile/Maggio 2017

PREISTORIA E STORIA

aleph	beth	gimel	daleth	he	waw	zayin	heth	teth	yod	kaph
א	ב	ג	ד	ה	ו	ז	ח	ט	י	כ
taw	sin shin	resh	qoph	sadhe	pe	ayin	samkeh	nun	mem	lamed
ת	ש	ר	ק	צ	פ	ע	ס	נ	מ	ל

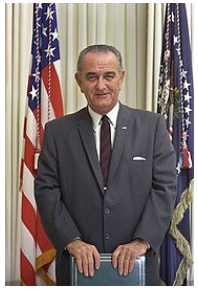
A	B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	V	X
D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	V	X	A	B	C

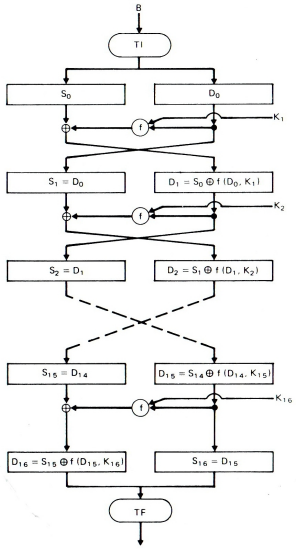
P	A	R	I	S	V	A	U	T	B	I	E	N	U	N	E	M	E	S	S	E
L	O	U	P	L	O	U	P	L	O	U	P	L	O	U	P	L	O	U	P	L

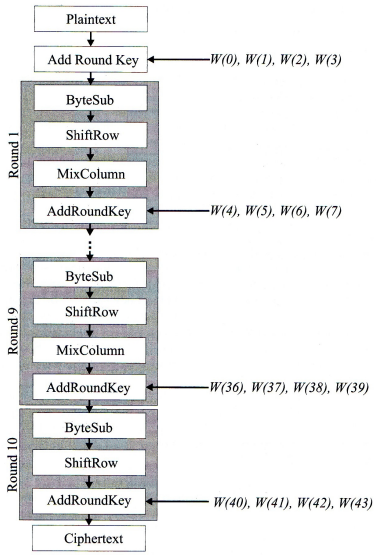


PREISTORIA E STORIA

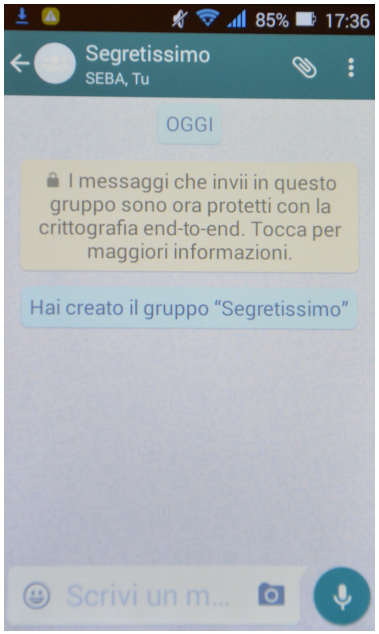




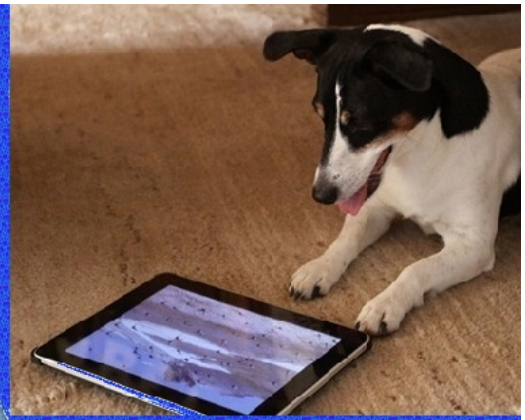




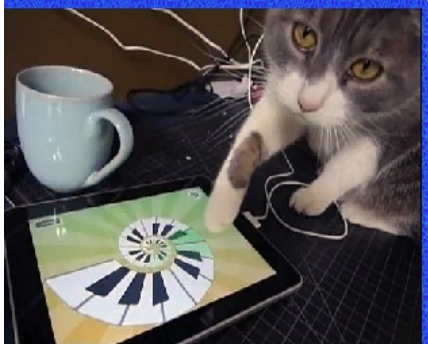
PREISTORIA E STORIA



Crittografia informatica






Crittografia informatica

- L'informazione è digitalizzata
- Il codice è un algoritmo
- Cos'è un algoritmo?
- Il primo a darne la definizione "rigorosa" fu Turing, nel famoso articolo del 1936
- Ci sono differenze tra algoritmo e programma? Un sistema operativo è un algoritmo? ...

Crittografia informatica

- L'informazione è digitalizzata
- Il codice è un algoritmo
- Cos'è un algoritmo?
- Il primo a darne la definizione "rigorosa" fu Turing, nel famoso articolo del 1936
- Ci sono differenze tra algoritmo e programma? Un sistema operativo è un algoritmo? ...

Crittografia informatica

- L'informazione è digitalizzata
- Il codice è un algoritmo
- Cos'è un algoritmo?
- Il primo a darne la definizione "rigorosa" fu Turing, nel famoso articolo del 1936
- Ci sono differenze tra algoritmo e programma? Un sistema operativo è un algoritmo? ...

Crittografia informatica

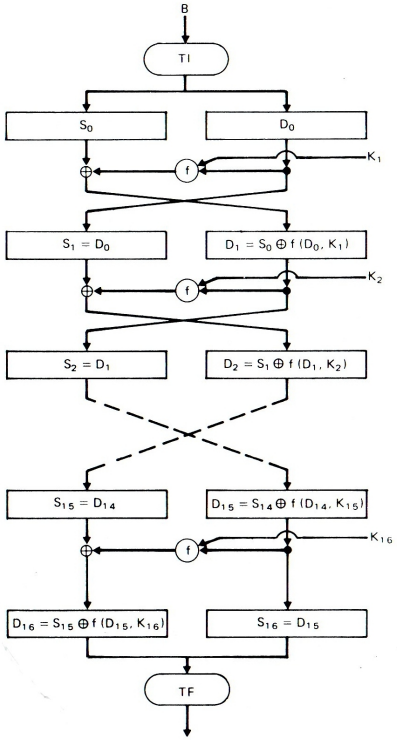
- L'informazione è digitalizzata
- Il codice è un algoritmo
- Cos'è un algoritmo?
- Il primo a darne la definizione "rigorosa" fu Turing, nel famoso articolo del 1936
- Ci sono differenze tra algoritmo e programma? Un sistema operativo è un algoritmo? ...

Crittografia informatica

- Si deve assumere il principio diKerckhoffs nella sua forma più stringente: l'algoritmo per la cifrazione/decifrazione dev'essere pubblico.
- In pratica, è a disposizione di chiunque (in rete) un *codice sorgente* (p.es., un programma C) che implementa la codifica (e uno per la decodifica, eventualmente lo stesso)
- La chiave (meglio se non troppo lunga) va tenuta invece segreta.
- Il codice dev'essere attaccabile solo dalla *forza bruta* (= niente scorciatoie intelligenti come quelle viste la scorsa lezione. Dobbiamo provare tutte le chiavi) e anche mettendo assieme tutti i calcolatori del mondo il tempo necessario ad applicare la forza bruta dev'essere disarmante!
- Dev'essere veloce (comunicazioni riservate anche telefoniche)!

Data Encryption Standard

- 1973 il National Bureau of Standards (ora NIST) richiede un algoritmo standard di cifratura
- 1975 IBM (Horst Feistel et al) definisce il codice che nel 1976/77 diventa DES (Data Encryption Standard)
- Il funzionamento del DES è pubblico (soddisfa il principio di Kerckhoffs).
- Del DES tutto è noto tranne la chiave, costituita da 8 bytes, ove un bit per byte è un bit di controllo (desumibile dalla chiave).
- Dunque la vera lunghezza della chiave è 56 bits (spazio di $2^{56} \approx 7.2 \times 10^{16}$).



Data Encryption Standard

Qualche numero e la fine del DES

- Malgrado anni e anni di studi non si arrivò mai a metodi più intelligenti della forza bruta
- $2^{56} \approx 7.2 \cdot 10^{16}$
- Supponiamo di saper testare una chiave in $1\mu s$
- Dunque sapremmo verificare $10^6 \cdot 60 \cdot 60 \cdot 24 = 8.64 \cdot 10^{10}$ chiavi al giorno.
- Farebbero circa 834 000 giorni (2285 anni)
- Ma se avessimo 100 000 calcolatori, basterebbero 8 giorni!
- In seguito a una competizione, Rocke Verser preparò un programma in grado di distribuire il lavoro su PC vari collegati in Internet. Nel 1997 furono necessari 5 mesi (ovviamente con carico piuttosto irregolare) a forzare il DES. Nel 1998 bastarono 39 giorni.

Data Encryption Standard

Qualche numero e la fine del DES

- Malgrado anni e anni di studi non si arrivò mai a metodi più intelligenti della forza bruta
- $2^{56} \approx 7.2 \cdot 10^{16}$
- Supponiamo di saper testare una chiave in $1\mu s$
- Dunque sapremmo verificare $10^6 \cdot 60 \cdot 60 \cdot 24 = 8.64 \cdot 10^{10}$ chiavi al giorno.
- Farebbero circa 834 000 giorni (2285 anni)
- Ma se avessimo 100 000 calcolatori, basterebbero 8 giorni!
- In seguito a una competizione, Rocke Verser preparò un programma in grado di distribuire il lavoro su PC vari collegati in Internet. Nel 1997 furono necessari 5 mesi (ovviamente con carico piuttosto irregolare) a forzare il DES. Nel 1998 bastarono 39 giorni.

Data Encryption Standard

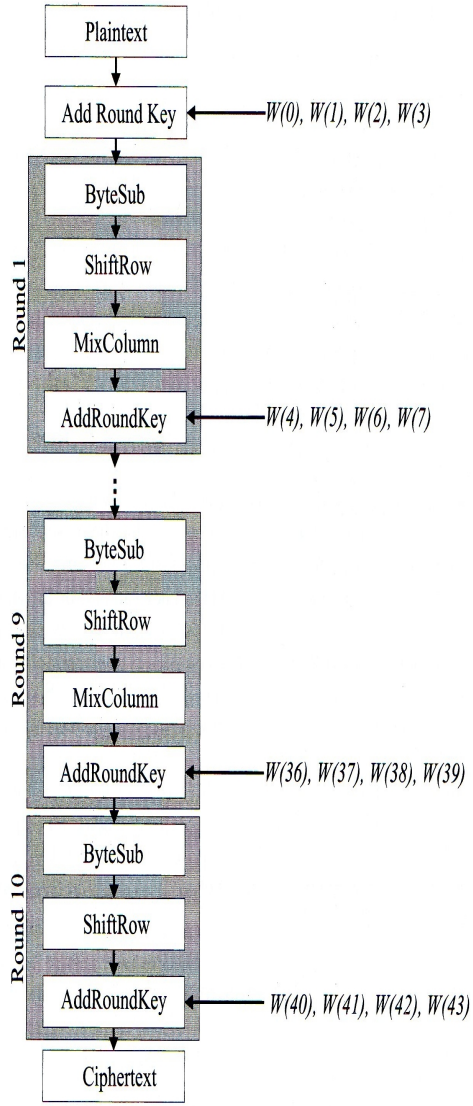
Qualche numero e la fine del DES

- Malgrado anni e anni di studi non si arrivò mai a metodi più intelligenti della forza bruta
- $2^{56} \approx 7.2 \cdot 10^{16}$
- Supponiamo di saper testare una chiave in $1\mu s$
- Dunque sapremmo verificare $10^6 \cdot 60 \cdot 60 \cdot 24 = 8.64 \cdot 10^{10}$ chiavi al giorno.
- Farebbero circa 834 000 giorni (2285 anni)
- Ma se avessimo 100 000 calcolatori, basterebbero 8 giorni!
- In seguito a una competizione, Rocke Verser preparò un programma in grado di distribuire il lavoro su PC vari collegati in Internet. Nel 1997 furono necessari 5 mesi (ovviamente con carico piuttosto irregolare) a forzare il DES. Nel 1998 bastarono 39 giorni.

Advanced Encryption Standard

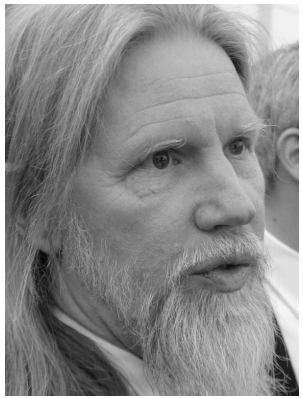
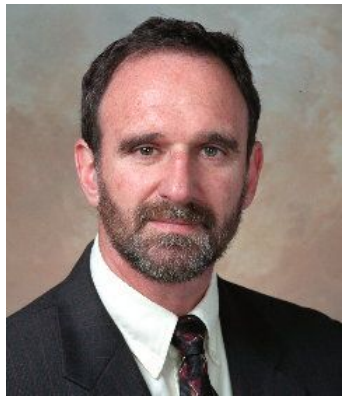
- L'algoritmo Rijndael di Joan Daemen and Vincent Rijmen fu annunciato dalla NIST come standard (AES) nel 2002

- Come per il DES, il funzionamento è totalmente pubblico.
- AES è in tre versioni differenziate dalla lunghezza della chiave:
  AES-128, AES-192 e AES-256.
- Ha ancora il problema dello scambio delle chiavi, però!

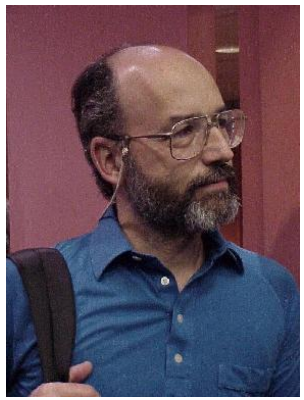
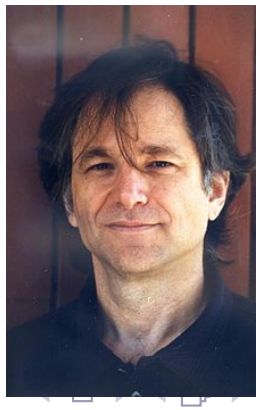


Crittografia a chiave pubblica

Idea “astratta” di Whitfield Diffie e Martin Hellman nel 1976 — Turing award nel 2016 (prima di loro M.J. Williamson, J.H. Ellis, C.C. Cocks $\simeq$ 70 del *Government Communications Head Quarter (GB)*)

Realizzazione concreta (algoritmo “difficile”) nel 1978 di Ronald Rivest, Adi Shamir e Leonard Adleman (RSA) — Turing award nel 2002

Alice

Bob

Charlie (la spia)

- Alice e Bob (e tutti) pubblicano (in un elenco telefonico, negli anni '70, in rete oggi) una volta per sempre la loro chiave pubblica (K_A quella di Alice, K_B quella di Bob, etc.), nota a tutti.
- Alice vuole inviare il messaggio m a Bob.
- Alice codifica il messaggio m per Bob con la chiave pubblica di Bob (K_B) e invia il messaggio cifrato $COD(m, K_B)$.
- Bob riceve il messaggio $COD(m, K_B)$ e usa la sua chiave privata H_B per decodificare il messaggio.

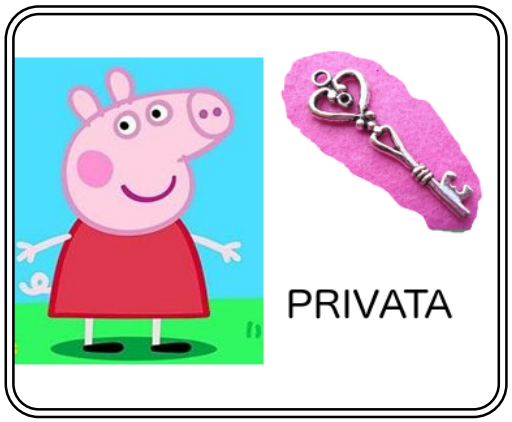
- Le chiavi pubblica e privata sono progettate in modo tale che

$$DEC(COD(m, K_B), H_B) = m$$

- L'operazione di decifrazione, sapendo la chiave privata dev'essere algoritmicamente facile
- L'operazione di decrittazione (per Charlie) deve essere algoritmicamente impraticabile.
- Anche se l'impresa è possibile: conoscendo K_B e $c = COD(m, K_B)$ si possono generare uno ad uno i messaggi di lunghezza opportuna, $m_1, m_2, \ldots, m_\ell$.
- Dunque si codificano uno ad uno con la chiave K_B e si vede se $COD(m_i, K_B) = c$.

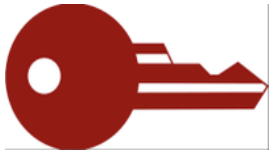
Crittografia a chiave pubblica

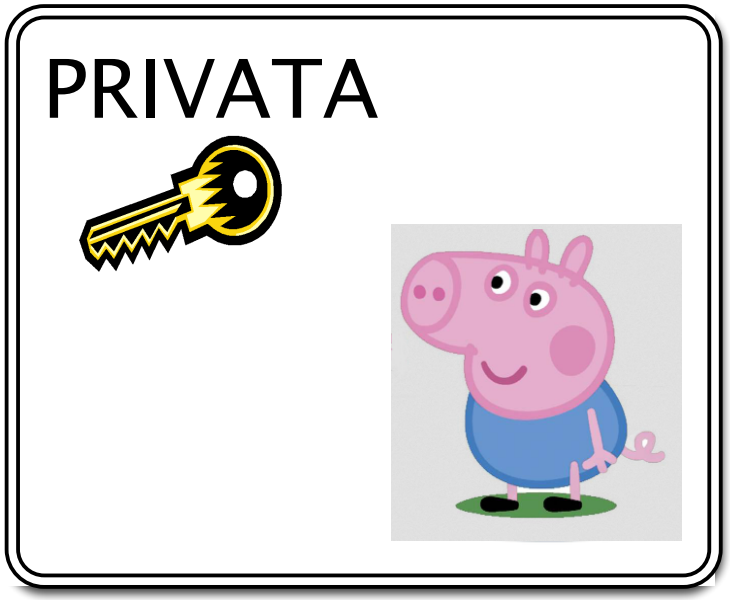
Peppa spedisce un messaggio a George



PUBBLICA

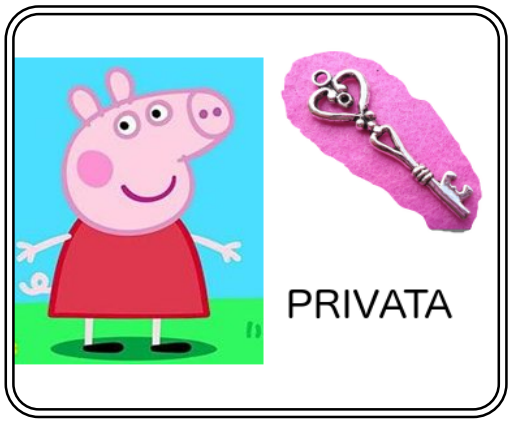
PUBBLICA





Crittografia a chiave pubblica

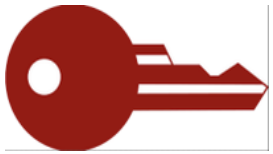
Peppa spedisce un messaggio a George



PUBBLICA

CIAO

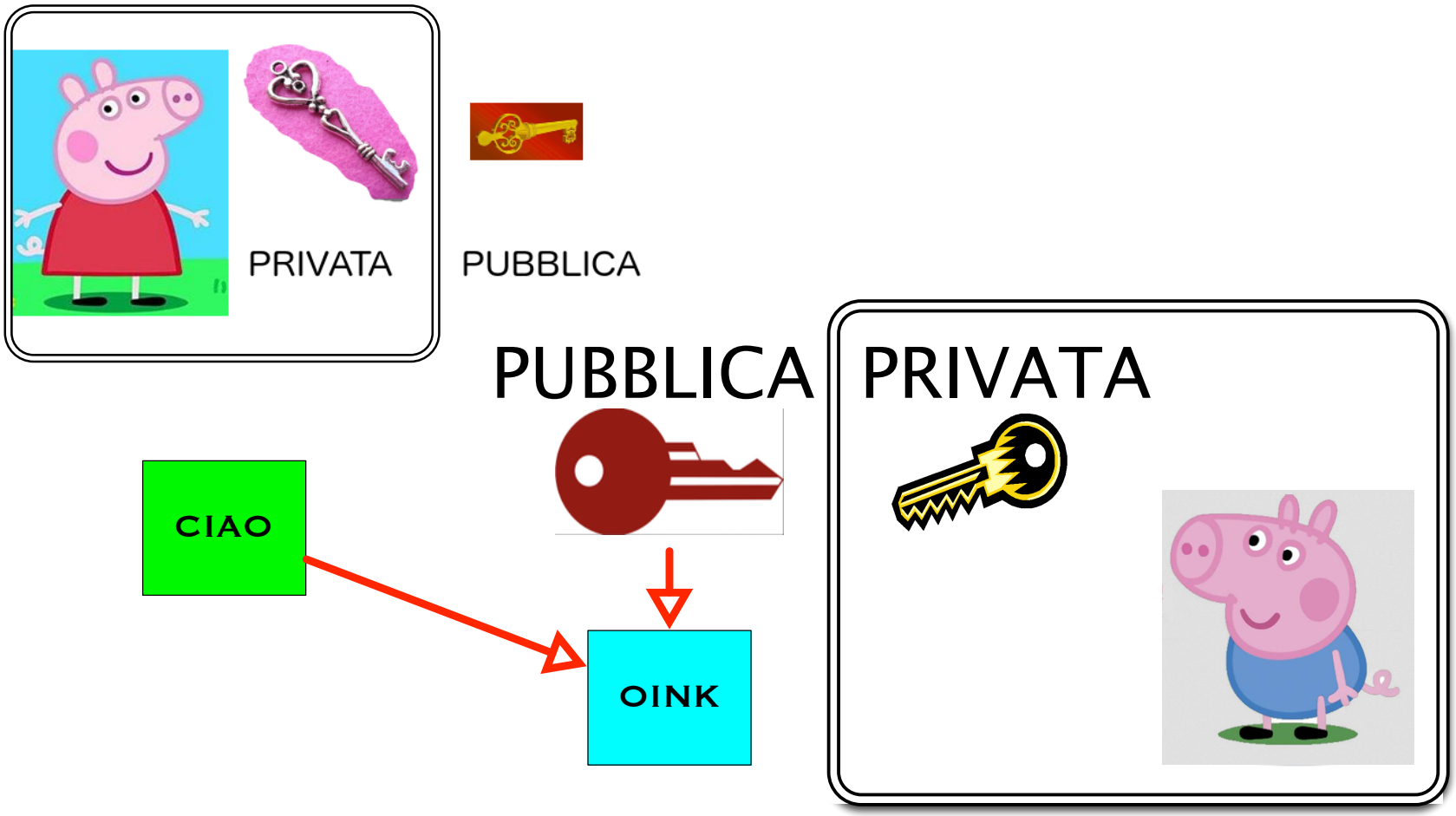
PUBBLICA



PRIVATA

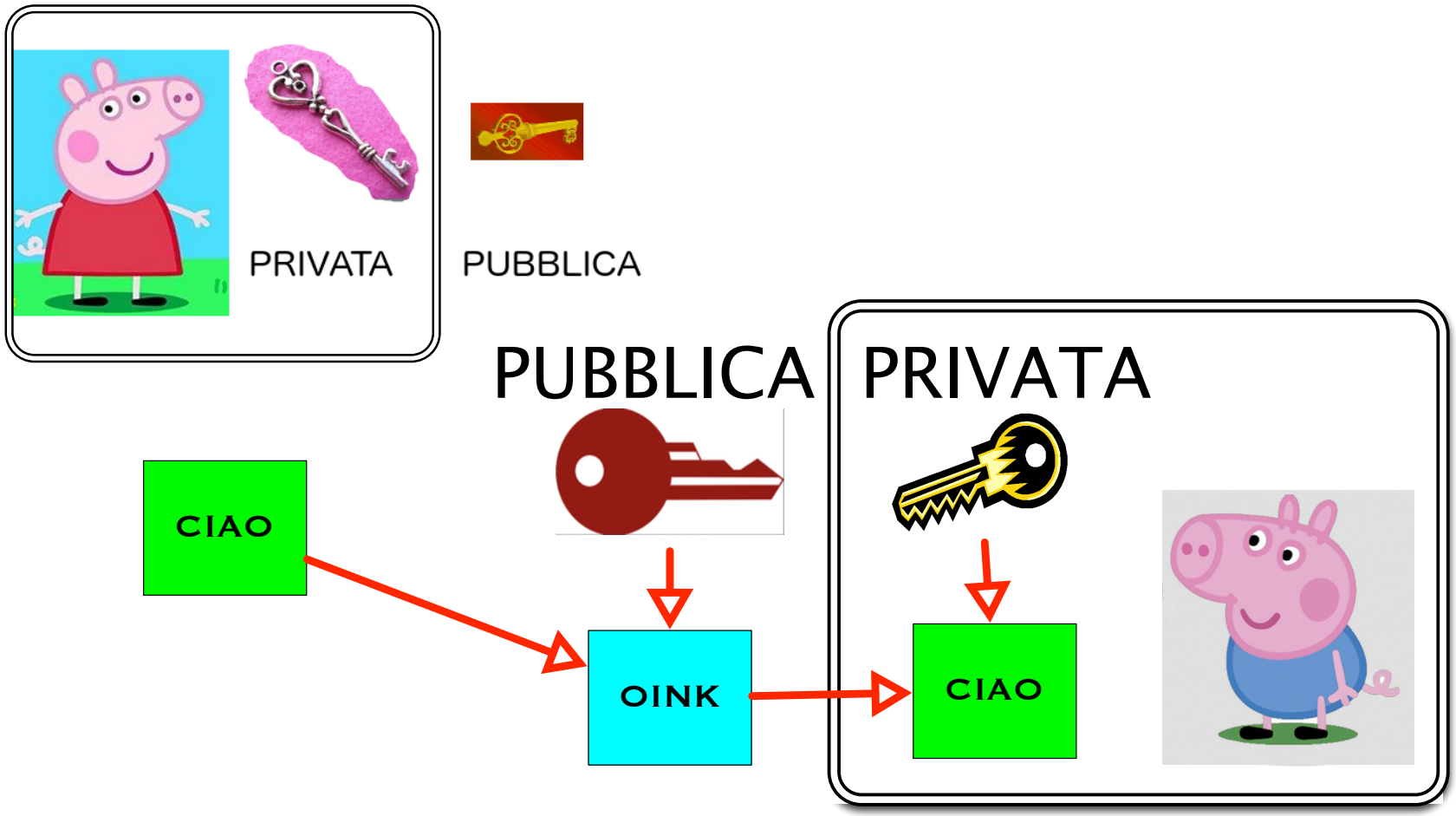
Crittografia a chiave pubblica

Peppa spedisce un messaggio a George



Crittografia a chiave pubblica

Peppa spedisce un messaggio a George



- L'operazione di decifrazione, sapendo la chiave privata dev'essere algoritmicamente facile
- L'operazione di decrittazione (per la spia) deve essere algoritmicamente impraticabile.
- Anche se l'impresa è possibile, avendo tempo a sufficienza: un modo è quello di provare tutti i messaggi di una data lunghezza, codificarli con la chiave pubblica di George, quando inseriamo CIAO, troveremo OINK: abbiamo capito che Peppa ha scritto CIAO.
- Così si scopre il messaggio ma non la chiave privata.

Bell'idea, ma come si realizza in pratica???

1. Bob sceglie due numeri primi p e q.
2. Verificare se un numero è primo (o meno) è algoritmicamente *facile* (pertanto, Bob prende un numero dispari delle dimensioni desiderate, ...)
3. Bob calcola $n = pq$ (e altre cose che omettiamo qui per semplicità) e rende n pubblico.
4. Alice vuole inviare un messaggio a Bob, conoscendo la chiave pubblica di Bob n. Lo spezza in blocchi di dimensione opportuna. Sia m un blocco: calcola $c = COD(m, n)$ (algoritmicamente *facile*).
5. Bob conoscendo p e q è in grado di calcolare $m = DEC(c, p, q)$ (algoritmicamente *facile*).
6. Charlie intercetta c sa che Alice l'ha inviato a Bob e conosce n pubblicati da Bob.
7. Al solito, in principio, può generare tutti i messaggi $m_1, m_2, \ldots, m_t$ di lunghezza opportuna (sono in numero finito, dato n), effettuare la codifica e vedere se trova c (forza bruta). È impraticabile.

PROBLEMA:

CHARLIE VORREBBE CALCOLARE p E q SAPENDO $n = pq$.

trova un fattore di n:
prova i da 2 a $\sqrt{n}$
se n diviso i ha resto 0
allora stampa i e finisci
(altrimenti aumenta i)

Nel caso peggiore compie un numero di passi pari a $\sqrt{n}$.
Possiamo togliere i pari, i multipli di 3, i multipli di 5, ma la sostanza non cambia.
Se n è un numero di 100 cifre, ogni divisione mi costa $10^{-10}s$, e dispongo di 10^9 processori, mi serve tempo

$$\approx \frac{10^{-10}\sqrt{10^{100}}}{3600 \cdot 24 \cdot 365 \cdot 10^9} = 3 \cdot 10^{23} \text{ANNI}$$

PROBLEMA:

CHARLIE VORREBBE CALCOLARE p E q SAPENDO $n = pq$.

trova un fattore di n:
prova i da 2 a $\sqrt{n}$
se n diviso i ha resto 0
allora stampa i e finisci
(altrimenti aumenta i)

Nel caso peggiore compie un numero di passi pari a $\sqrt{n}$.
Possiamo togliere i pari, i multipli di 3, i multipli di 5, ma la sostanza non cambia.

Se n è un numero di 100 cifre, ogni divisione mi costa $10^{-10}s$, e dispongo di 10^9 processori, mi serve tempo

$$\approx \frac{10^{-10}\sqrt{10^{100}}}{3600 \cdot 24 \cdot 365 \cdot 10^9} = 3 \cdot 10^{23} \text{ANNI}$$

PROBLEMA:

CHARLIE VORREBBE CALCOLARE p E q SAPENDO $n = pq$.

trova un fattore di n:
  prova i da 2 a $\sqrt{n}$
    se n diviso i ha resto 0
    allora stampa i e finisci
    (altrimenti aumenta i)

Nel caso peggiore compie un numero di passi pari a $\sqrt{n}$.
Possiamo togliere i pari, i multipli di 3, i multipli di 5, ma la sostanza non cambia.
Se n è un numero di 100 cifre, ogni divisione mi costa $10^{-10}s$, e dispongo di 10^9 processori, mi serve tempo

$$\approx \frac{10^{-10}\sqrt{10^{100}}}{3600 \cdot 24 \cdot 365 \cdot 10^9} = 3 \cdot 10^{23} \text{ANNI}$$

PROBLEMA:

CHARLIE VORREBBE CALCOLARE p E q SAPENDO $n = pq$.

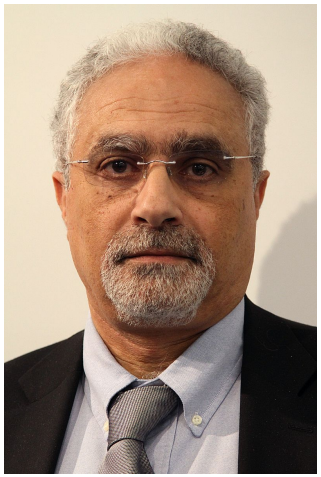
trova un fattore di n:
  prova i da 2 a $\sqrt{n}$
    se n diviso i ha resto 0
    allora stampa i e finisci
    (altrimenti aumenta i)

Nel caso peggiore compie un numero di passi pari a $\sqrt{n}$.
Possiamo togliere i pari, i multipli di 3, i multipli di 5, ma la sostanza non cambia.
Se n è un numero di 100 cifre, ogni divisione mi costa $10^{-10}s$, e dispongo di 10^9 processori, mi serve tempo

$$\approx \frac{10^{-10}\sqrt{10^{100}}}{3600 \cdot 24 \cdot 365 \cdot 10^9} = 3 \cdot 10^{23}\text{ANNI}$$

ELGAMAL E IL LOGARITMO DISCRETO

- Nel 1985 Taher Elgamal architetta un altro sistema di cifratura a chiave pubblica.
- La *forza* del sistema di cifratura è basata sulla difficoltà computazionale di calcolare il logaritmo in un campo finito (in breve *logaritmo discreto*).



Ometto i dettagli tecnici di codifica e decodifica

- Consideriamo l'insieme $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$ in cui usiamo l'aritmetica modulo p. Ovvero, se il risultato a di una operazione esce dall'insieme prendiamo il resto della divisione tra a e p.
- Ad esempio, in $\mathbb{Z}_7$,

$+$	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

- Si noti che ogni numero ha un opposto: ad esempio, poiché 4 + 3 = 0, si ha che 4 = -3 e 3 = -4.

- Sempre in $\mathbb{Z}_7$,

$*$	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	**1**	2	3	4	5	6
2	0	2	4	6	**1**	3	5
3	0	3	6	2	5	**1**	4
4	0	4	**1**	5	2	6	3
5	0	5	3	**1**	6	4	2
6	0	6	5	4	3	2	**1**

- Si noti che ogni numero ha (esattamente) un inverso: ad esempio, poiché $3 * 5 = 1$, si ha che $3 = 5^{-1}$ e $5 = 3^{-1}$.

- Le proprietà accennate (ovvero, essere un *campo finito*) valgono per $\mathbb{Z}_p$ quando p è un numero primo.
- Possiamo dunque definire l'elevamento a potenza
  $3^1 = 3$ $3^2 = 9 \equiv_7 2$, $3^3 = 27 \equiv_7 6$,
  $3^4 = 81 \equiv_7 4$ $3^5 = 3^4 \cdot 3 \equiv_7 4 \cdot 3 \equiv_7 5$ $3^6 = 3^5 \cdot 3 \equiv_7 5 \cdot 3 \equiv_7 1$
- Cosideriamo ora due numeri α e x (numeri tra 1 e $p-1$, α meglio se radice primitiva come 3 nell'esempio sopra).
- Calcoliamo $\beta = \alpha^x$ modulo p (non è difficile farlo in modo efficiente).
- Dati α, β e p, trovare x è possibile ma, ancora, computazionalmente impraticabile (se p è un numero di 200 cifre o più). $x = \log_\alpha \beta$ in $\mathbb{Z}_p$.

Riassunto

- Abbiamo visto il punto di arrivo della crittografia “tradizionale”, a chiave segreta scambiata privatamente: AES. Pros: Cifratura e decifratura velocissima. Cons: Scambio della chiave.
- Abbiamo visto l’idea dei cifrari a chiave pubblica e due implementazioni
  - RSA $\Rightarrow$ fattorizzazione
  - ElGamal $\Rightarrow$ Logaritmo Discreto

  (ce ne sono altre che elaborano ancora le idee come i codici ellittici) Pros: diffuzione pubblica della chiave. Cons: cifratura e decifratura lentina (specie pensando a invio di audio/video etc)
- Possiamo combinare gli effetti positivi e annullare quelli negativi?

Un ulteriore passo avanti

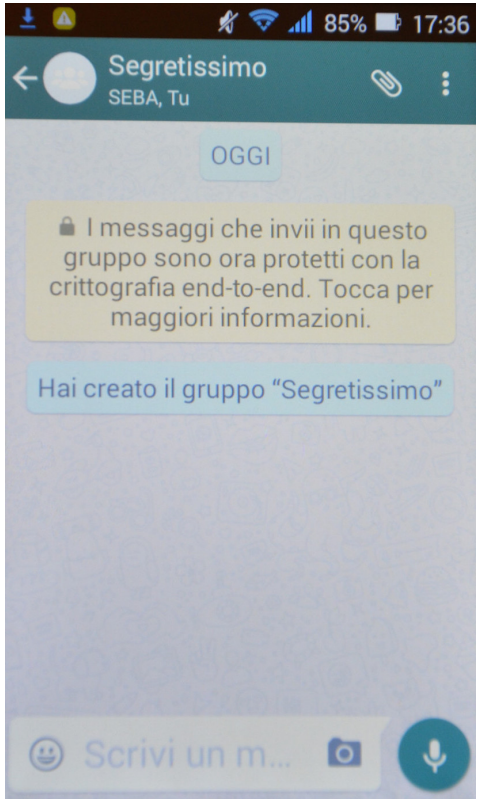
PGP e la sicurezza delle email

- Nel 1991Phil R. Zimmerman sviluppa PGP (Pretty Good Privacy)
- Cifrari a chiave privata (AES, DES) e Chiave Pubblica (RSA, ElGamal) vengono usati in combinazione.
- Usando RSA ci si passa la chiave (corta) per l'AES, e si usa AES (più veloce) per cifrare il messaggio (lungo).

Zimmeman ebbe una grande idea, senza secondi fini. Fu sottoposto a diversi processi negli USA.

Conclusioni

La sicurezza . . . quotidiana



Siamo immersi in un mondo connesso e con molti malintenzionati.
La crittografia a chiave pubblica ci permette di comunicare in sicurezza.
Sempre che non si sappia fattorizzare n in p e q o calcolare $\log_\alpha \beta$ in $\mathbb{Z}_p$.