

3. Crittografia

Sicurezza dell'Informazione

Avvertimento

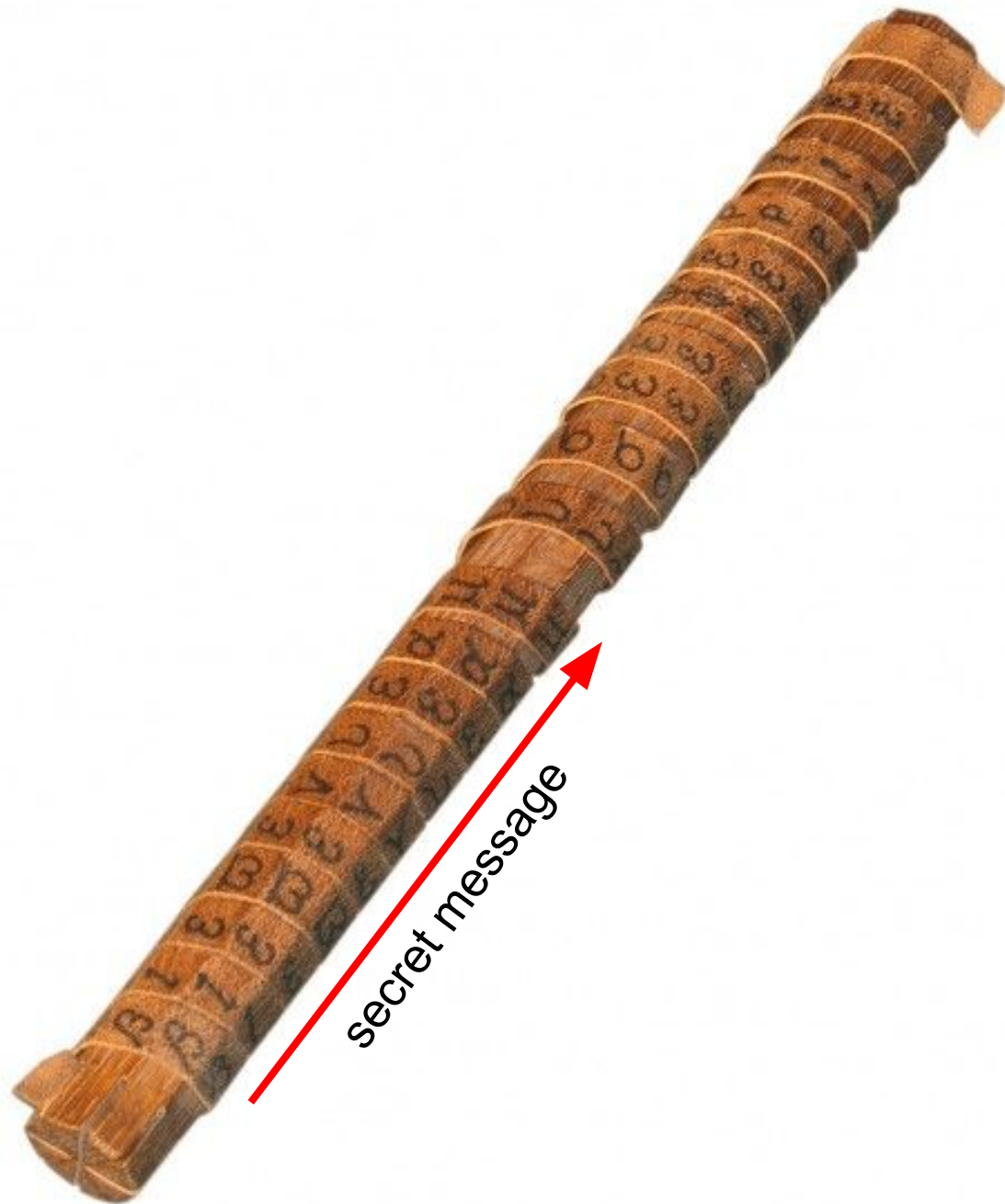
- Questa è una breve introduzione semplificata alla crittografia.
- Presenteremo solo ciò che è necessario per le discussioni sulla sicurezza dei sistemi.
- Nella maggior parte dei casi tratteremo i concetti matematici come delle “scatole nere”.

Un cenno storico

Dal greco: *kryptós* = “nascosto” e *gráphein* = “scrivere” (ossia “l’arte della scrittura segreta”).

Nell’antichità, la scrittura stessa era già una “tecnica segreta”.

La crittografia nasce nella società greca, quando la scrittura divenne più comune e si sentì il bisogno di sviluppare forme di scrittura nascosta.



Il cifrario di Cesare (o a rotazione)

m

C	R	Y	P	T	O	G	R	A	P	H	Y
2	17	24	15	19	14	6	17	0	15	7	24




chiave $k = 5$

c

7	22	3	20	24	19	11	22	5	20	12	3
H	W	D	U	Y	T	L	W	F	U	M	D

Cifrario a sostituzione

\mathcal{A}_m	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
\mathcal{A}_c	t	m	k	g	o	y	d	s	i	p	e	l	u	a	v	c	r	j	w	x	z	n	h	b	q	f



m C R Y P T O G R A P H Y



c k j q c x v d j t c s q

Pigpen

A	B	C
D	E	F
G	H	I

J .	K .	L .
M .	N .	O .
P .	Q .	R .

S
T **U**
V

W
X **Y**
Z

Cifrario di Vigenère (multi-rotazione)

m

C	R	Y	P	T	O	G	R	A	P	H	Y
2	17	24	15	19	14	6	17	0	15	7	24

chiave $k = \{5, 12, 7\}$



c

5	3	5	18	5	21	9	3	7	18	19	5
F	D	F	S	F	V	J	D	H	S	T	F

Kasisky Test

"OTFDWP DZ DJDG IMKDHIG ASSUKOE, NQYBZZVBI ZZRZWYVBXP NDPVZTNI JZMLCO, IN UW RCXVDKRDO NADN IXZOZJB DDDKDU WA UKB EEI, MJ RD UVDSUUA, EMGXDQXMJ, JDL KQRABA; VZU CXWPSY EQZDALB DIOUFWI PVHV RD AJYV FQG MQTXWVDLVM Q KZDKJYV MAPJB XMQVVYVZZTN YV OTZB XCZ; QMND BCQ SJHJVDZL, WZVZU XBL FUEPI WA BVPK XGMTRDO OTV CYBGQ "CXHL JR **KQU** ECUKN UTZBYJDBN" MSXLM VXC CXMDD FCXMM YRPDQGAHDUVO MJLHQKFZXDA JR UXCQIUFW; QVY **FYN** CWYQIW AQISJ XV ADMD DDNPDCRDO OTV BQUZ EEXM-ECUKN GCVPIDFMY UE CXM MAPJB AOMEMQZY; MEM **JPZ** TRWEDZDZJD NGMX KUIMUEP **JPZ** AEN VQBGIN EN V EEXM-ECUKN SPVDXNH; IIP **KQU** OMQRC QCNFIRQV ZYGRHM, XMVBQZDME, QUQM FF XLMMXFATQIS IXCM, CMMRDO AAI CXM DYGNHQVX TXBWM **FYN** IIHQ ZVFMMURU XCZ; MEM JPJGXQ JPDE GAU-MHUENDKZ UE RJ IKBCRUA OA **KQU** PPYRW HIXQ ZCIMGR, XRLQIS **KQU** ECUKN CII UUNQT HMJCUZNTZY EDZD VEUZT PLBAG ODZKU; IIP KQECBT, SNIQYQJ JBT OTZB, MPDFVWUAN TRB RMZZ VEUV HMUN IQBZZOYKVZK XV OGMUWUAN, RFA QUIZX CXM MADJDA V IYRJM NFFWU UVDBNT I EAPOKT YMP; JDL OTFDWP DZ FCXMM YFAJIG EPVFIOTZNI IIP JHCJJXZIYVBE, KQYA NMDN XCZ UJ VQLZ **FYN** UUWXVV EN HMEH JWPOYRDO, IASUU BCUEPI- BCQ ZWDWXQELU WA NIRTMN, **FYN** RMIUXWYBT AW JWM; OTFDWP VYFWW BCQ INT UZZ FO QUZDZLQ BCQ XRLQIS FO **JPZ** IYRJM WQCC EN RMDYKU RMJ CXM YQVYUAO BCNTOZ AW QEVDJ; KQECBT ZW CIIK TUYUZE, NQYBZZVBI BTBZOYMN **FYN** CIEQJCO WA VLBQXQ ZW **JPZ** QIVYVZ AW CXM EGUPU, IIP TXDBMUSDJMN FF CXM YMZUO AOMKN EN FUEPI IIP HDUMIE UAQEI NP VYTF-IYRJM NFNNTA"

Rivelava la lunghezza della chiave

Quando la matematica ha vinto la guerra

- Durante la Seconda Guerra Mondiale, Alan Turing lavorò a Bletchley Park per decifrare i codici delle potenze dell'Asse, in particolare il cifrario Enigma.
- La nascita dei primi computer universali fu in parte stimolata proprio da questo sforzo di decrittazione.



Definizione

È la pratica e lo studio delle tecniche che forniscono:

- **Confidenzialità:** solo le parti autorizzate possono accedere alle informazioni.
- **Integrità:** i dati non possono essere alterati o modificati senza essere rilevato.
- **Autenticazione:** verifica dell'identità delle parti coinvolte nella comunicazione.
- **Non ripudio:** impedisce a una parte di negare di aver inviato o ricevuto un messaggio.

Cosa ci serve

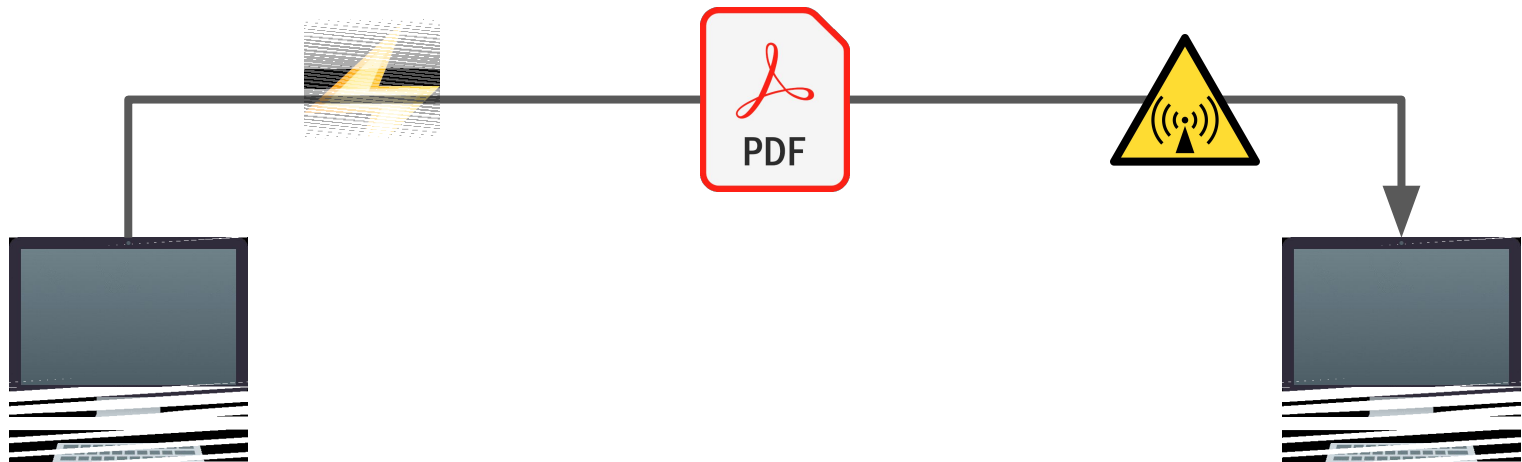
A noi interessa principalmente:

- **Funzioni di Hash:** per l'integrità dei dati.
- **Cifrari**
 - **simmetrici:** per comunicare velocemente con una sola chiave.
 - **asimmetrici:** per lo scambio di chiavi simmetriche e l'autenticazione.

Funzioni di Hash

Esempio: invio di file

Rumori di diversa natura possono causare alterazioni nei file che inviamo. Come ce ne accorgiamo?



Esempio: acquisizione di una prova digitale

Le fonti di prova (Hard disk e SSD) passano tramite diverse persone prima di arrivare in un tribunale:

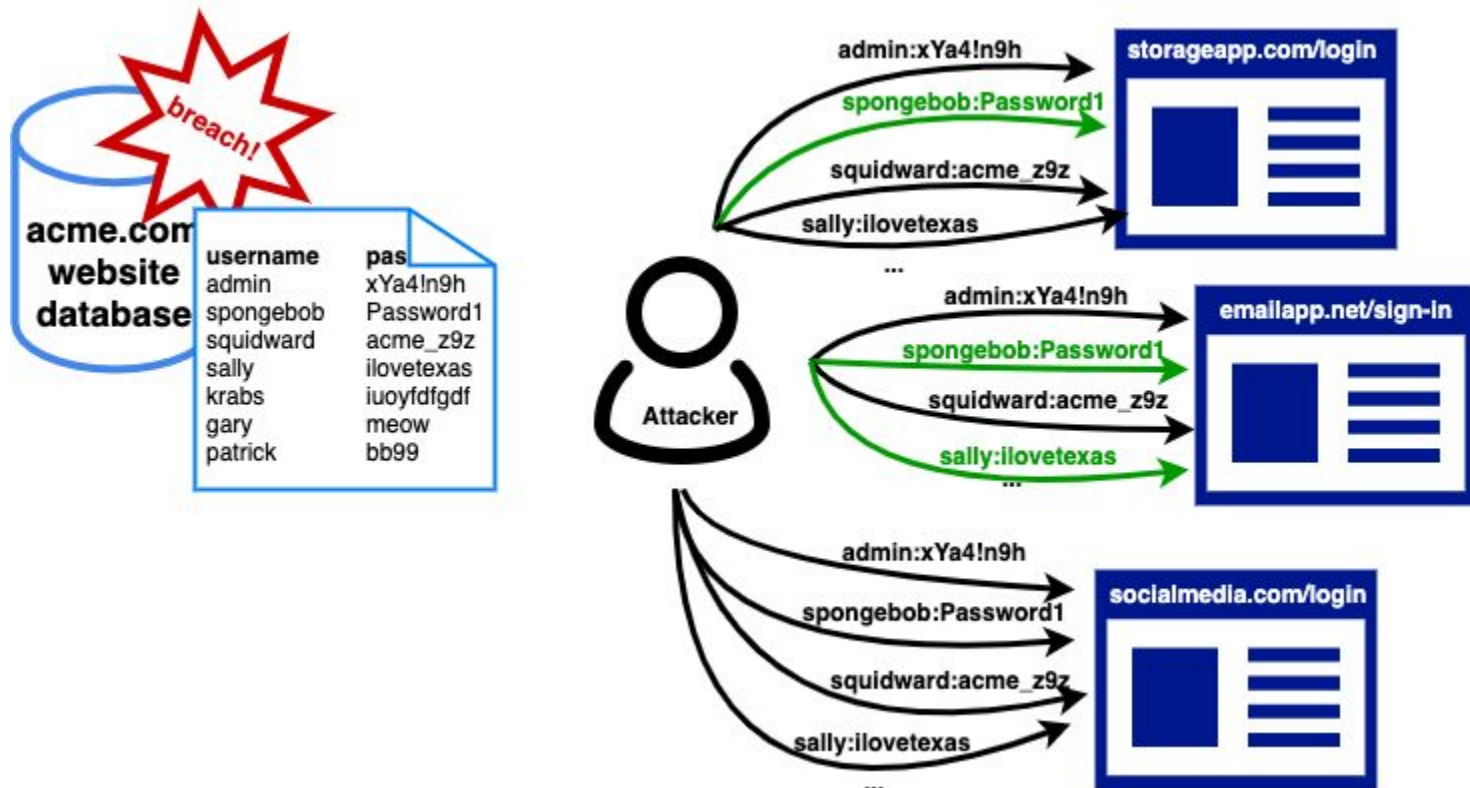
- Periti tecnici
- Supervisore delle indagini
- Analisti forensi
- Responsabili della custodia



Come sappiamo se è stata alterata?

Esempio: attacco informatico

Un hacker malevolo ruba le password da un sito. Molti utenti ri-usano le stesse password su altri siti. Possiamo anonimizzarle?



Idea generale

Ci serve uno strumento in grado di **verificare se qualcosa è cambiato**. Le funzioni di hash fanno questo.



*Dato un'entità digitale (binaria), la **funzione di hash** restituisce un numero (chiamato **digest**), che rappresenta quella specifica entità.*

Proprietà

Una funzione di hash deve essere:

- **Deterministica**: dato un input, restituisce sempre lo stesso digest.
- **Produrre un digest di lunghezza fissa (in bit)**: altrimenti risulterebbe complesso gestire diverse lunghezze.
- **Irreversibile**: dal digest non si deve poter risalire all'input.
- **Resistente alle collisioni**: deve essere difficile trovare 2 input con lo stesso digest.
- **Veloce da calcolare**: per ragioni di performance.

Una semplice funzione di hash

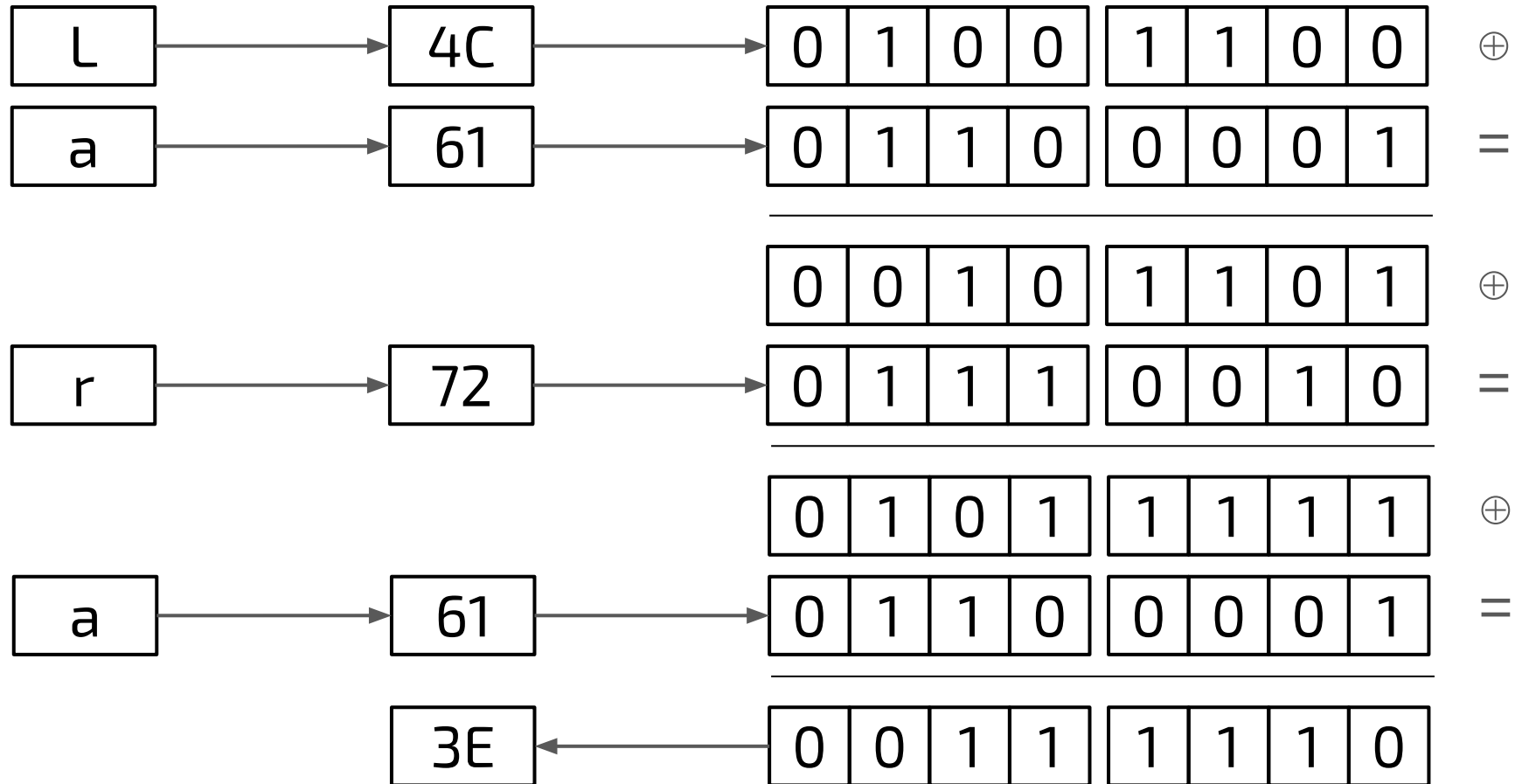
Un esempio di una funzione di hash è una funzione che prende tutti i byte e fa una xor fra di loro:

$$H(x) = x_1 \oplus x_2 \oplus x_3 \oplus \dots \oplus x_n$$

Esempio:

L	a	r	a
4C	61	72	61

Una semplice funzione di hash



Una semplice funzione di hash

Digest di “Lara” è 0x3E.

E quello di “Enzo”?

Una semplice funzione di hash

Digest di “Lara” è 0x3E.

E quello di “Enzo”?

Questa funzione di hash è:

- **Deterministica**
- **Produce un digest di lunghezza fissa (in bit)**
- **Irreversibile**
- ~~**Resistente alle collisioni**~~
- **Veloce da calcolare**

Resistenza alle collisioni

L'aspetto **più importante** di una funzione di hash è la **resistenza alle collisioni**. Senza questa caratteristica sarebbe facile:

- Non accorgersi di un cambiamento dovuto al rumore
- Manomettere le fonti di prova
- Proteggere le informazioni online

Funzioni di hash moderne

Algoritmo	Lunghezza (bit)	Digest di "ciao"
MD5	128 ($2^{128} \approx 3,4 \times 10^{38}$)	6e6bc4e49dd477ebc98ef4046c067b5f
SHA1	160 ($2^{160} \approx 1,46 \times 10^{48}$)	1e4e888ac66f8dd41e00c5a7ac36a32a9950d271
SHA256	256 ($2^{256} \approx 1,16 \times 10^{77}$)	b133a0c0e9bee3be20163d2ad31d6248db292aa6d cb1ee2d7fc0da29886a2a5d
SHA512	512 ($2^{512} \approx 1,34 \times 10^{154}$)	a0c299b71a9e59d5ebb07917e70601a3570aa103e 99a7b2c92e4b0fef2d8a6a26e2d64cd845c7f0fbc 7b383e4ac2a32d7f49b2911b0a09e301b78a35fd9 d69fc

Resistenza alle collisioni

Per resistenza alle collisioni si intende che è difficile trovare due input **qualsiasi** con lo stesso digest.

Casa	634e888a
Cucina	0e9ba298
Mobile	a2a86fef
Garage	d7231f49
Radio	0e9ba298
Microfono	d64cd845

Resistenza alla prima pre-immagine

Dato un digest ***d***, deve essere difficile trovare un input ***i*** tale che **$H(i) = h$** .

Digest: 1e4e888a

Deve essere difficile trovare un input ***i***, per esempio “segreto”, che da un digest ***d*** pari a 1e4e888a

Resistenza alla seconda pre-immagine

Dato un input i_1 , deve essere difficile trovare un input diversi i_2 tale che $H(i_1) = H(i_2)$.

Input: "Pippo" -> **Digest:** 4e478089

Deve essere difficile trovare un input i_2 che da lo stesso digest **d** di "Pippo" (ovvero 4e478089 nell'esempio)

Resistenza e tempi medi

Algoritmo	Collisione	Pre-Immagine
MD5	secondi	10^{18} anni
SHA1	giorni	10^{28} anni
SHA256	10^{32} anni	10^{57} anni
SHA512	10^{66} anni	10^{134} anni

Resistenza e tempi medi

Algoritmo	Collisione	Pre-Immagine
MD5	secondi	10^{18} anni
SHA1	giorni	10^{28} anni
SHA256	10^{32} anni	10^{57} anni
SHA512	10^{66} anni	10^{134} anni

MD5 e SHA1 sono oggi considerati insicuri

GDPR: Password e sicurezza online

Articolo 5, par. 1, lett. f) — Integrità e riservatezza

I dati personali devono essere trattati in modo da garantire un'adeguata sicurezza, compresa la protezione contro il trattamento non autorizzato o illecito e contro la perdita, la distruzione o il danno accidentale, mediante misure tecniche e organizzative adeguate.

GDPR: Password e sicurezza online

Se si vuole essere compliant con il GDPR, occorre proteggere/anonimizzare le password. L'hashing è la soluzione.

Username	Password Hash
Law	1e4e888a
Frank	0e9bee3b
Matteo	2e4b0fef
Rob	32d7f49b
Gab	e2d7fc0d
Tom	d64cd845

GDPR: Password e sicurezza online

Se l'attaccante riesce ad accedere al database le nostre password non sono del tutte compromesse. A meno che...

Username	Password Hash
Law	1e4e888a
Frank	0e9bee3b
Matteo	2e4b0fef
Rob	32d7f49b
Gab	e2d7fc0d
Tom	d64cd845

GDPR: Password e sicurezza online

Lookup Table

Username	Password Hash
Law	1e4e888a
Frank	0e9bee3b
Matteo	2e4b0fef
Rob	32d7f49b
Gab	e2d7fc0d
Tom	d64cd845

... l'attaccante non genera i digest di un dizionario di parole.
Nel gergo **Lookup Table**.

segreto	634e888a
password	1e4e888a
123456	a2a86fef
asdasd	d7231f49
mamma	0e9ba298
lorenzo	d64cd845
2004	e2d7fc0d
viadei...	683dea23
capricorno	54fe1082
rocky	232aedf2
milan	12543dee

Soluzione: salting

Alle password vengono concatenati dei caratteri casuali e poi viene calcolato l'hash.

Username	Salt	Password Hash
Law	3x31LwpZ	1e4e888a
Frank	2wV9q4w8	0e9bee3b
Matteo	iJ9xG2W8	2e4b0fef
Rob	l6T8u189	32d7f49b
Gab	94H0L1tV	e2d7fc0d
Tom	3m8HytT5	d64cd845

Soluzione: salting

L'attaccante deve creare una Lookup Table per utente -> **Impraticabile**

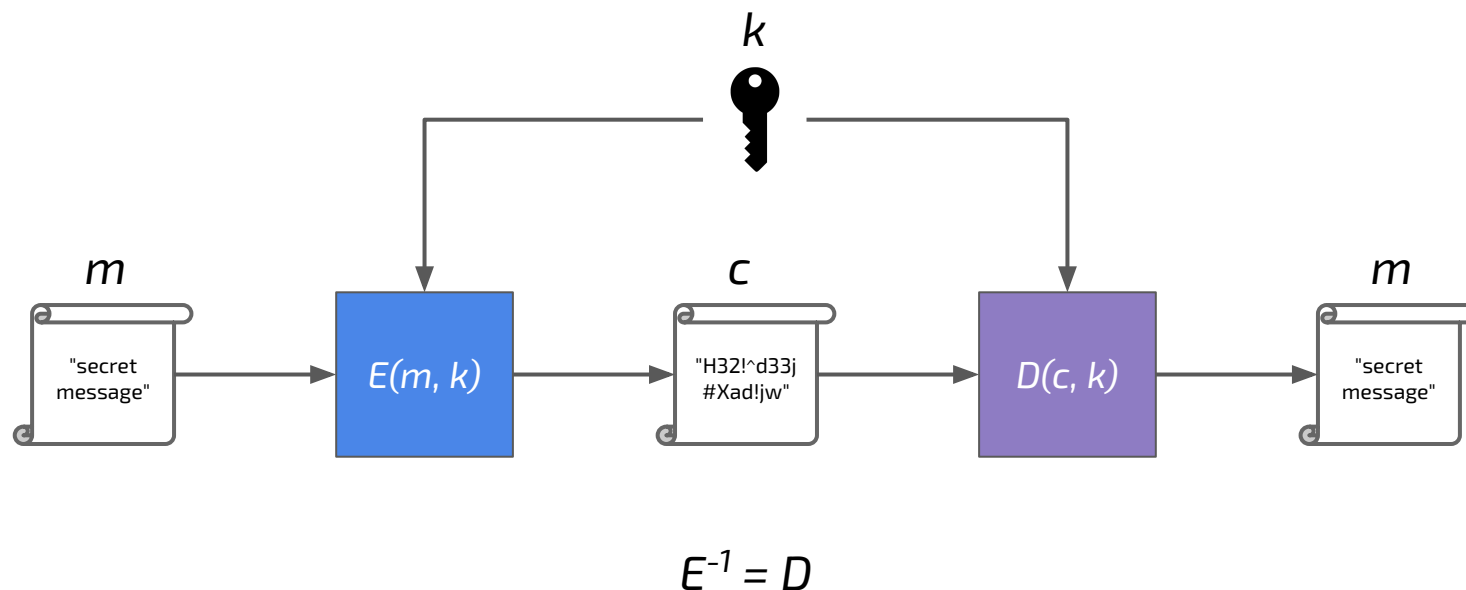
Username	Salt	Password Hash
Law	3x31LwpZ	2e4b0fef
Frank	2wV9q4w8	0e9bee3b
Matteo	iJ9xG2W8	32d7f49b
Rob	16T8u189	34fe7324
Gab	94H0L1tV	e2d7fc0d
Tom	3m8HytT5	d64cd845

Cifrari

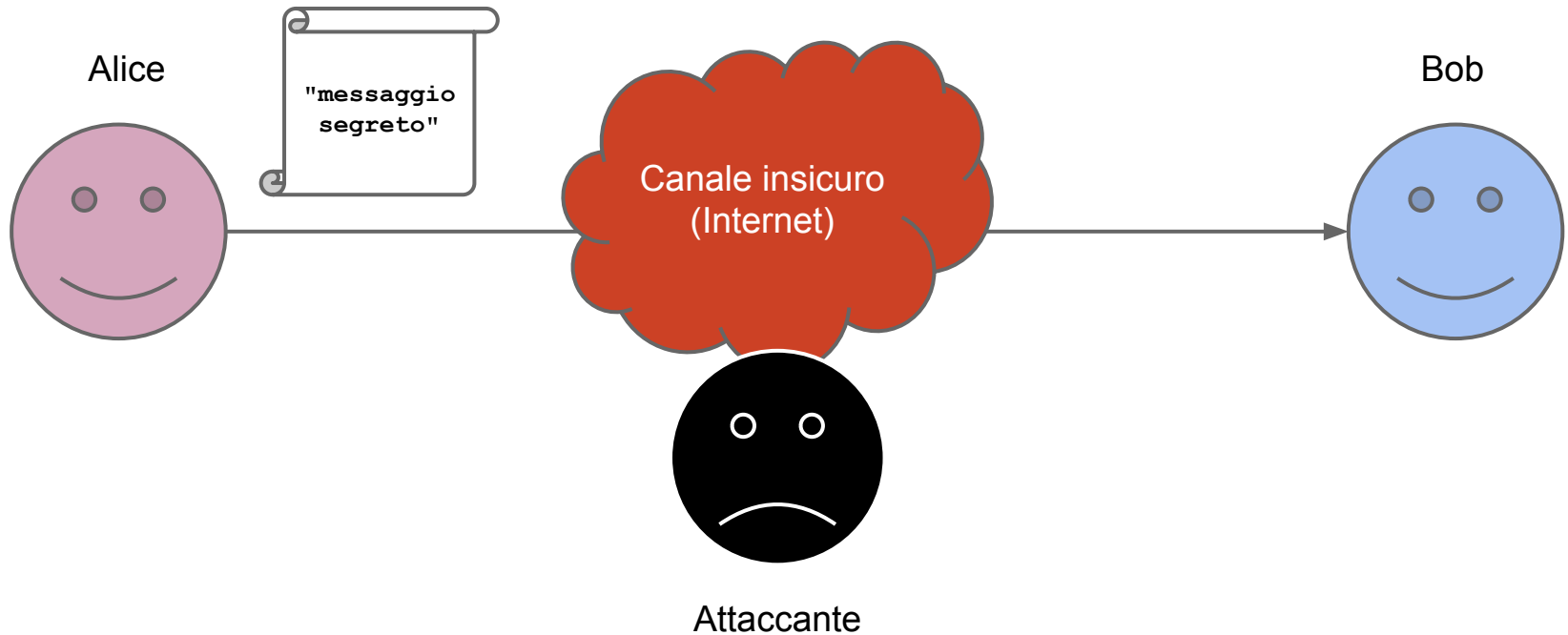
Crittosistema

Un crittosistema è un sistema composto da algoritmi crittografici.

Quando il suo obiettivo è garantire la riservatezza, esso prende in ingresso un messaggio (detto testo in chiaro) e lo trasforma in un testo cifrato mediante una funzione reversibile e una chiave.



Il problema da risolvere: confidenzialità



Il principio di Kerckhoffs

La sicurezza di un sistema crittografico deve basarsi solo sulla segretezza della chiave, e mai sulla segretezza dell'algoritmo.

Auguste Kerckhoffs, "La cryptographie militaire", 1883

Questo significa che:

- In un sistema crittografico sicuro non è possibile ricavare il testo in chiaro dal testo cifrato senza conoscere la chiave.
- Inoltre, non è possibile determinare la chiave analizzando coppie di testo in chiaro e testo cifrato.
- Gli algoritmi devono sempre essere considerati noti all'attaccante.

Il teorema di Shannon (1949)

Shannon definisce una cifratura perfetta come un sistema in cui:

conoscere il testo cifrato non dà alcuna informazione sul testo in chiaro.

In un cifrario perfetto, il numero di chiavi $|K|$ deve essere maggiore o uguale al numero di messaggi possibili $|M|$

$$|K| \geq |M|$$

Osservazione: se mando due volte lo stesso messaggio con la stessa chiave, rivelo una informazione.

Il cifrario perfetto: One-Time Pad (OTP)

- XOR di un messaggio m con una chiave casuale k della stessa lunghezza di m :

$$\text{lunghezza}(k) \geq \text{lunghezza}(m)$$

- La chiave è pre-condivisa e viene consumata durante la scrittura.
 - Non può mai essere riutilizzata!
- L'OTP (One-Time Pad) è un cifrario perfetto minimale:
 - minimale perché $|K| = |M|$
- Ma terribilmente scomodo, usato solo in contesti speciali (es. comunicazioni diplomatiche o militari ad alta sicurezza).

Imperfezione e brute force

- Gli algoritmi che usiamo quotidianamente sono imperfetti, e possono essere violati:
 - ogni coppia messaggio cifrato/chiaro rivela una piccolissima informazione (perché la chiave viene riutilizzata).
- L'unica cosa che non si conosce è la chiave (Kerckhoffs)
 - Ricordate: l'algoritmo è considerato noto.
- Brute forcing è possibile per ogni algoritmo di cifratura (non perfetto) moderno.
 - Si prova ogni possibile chiave fino a quando una produce un messaggio in chiaro che "ha senso".
- **Cifrari perfetti (es., one-time pad) non sono vulnerabili al brute force.**
 - Provare tutte le chiavi (randomiche) produrrà tutti i possibili messaggi in chiaro.

Crittoanalisi: violare i cifrari

Un crittosistema imperfetto è insicuro se esiste un modo di violarlo più veloce del brute forcing.

Tipi di attacchi:

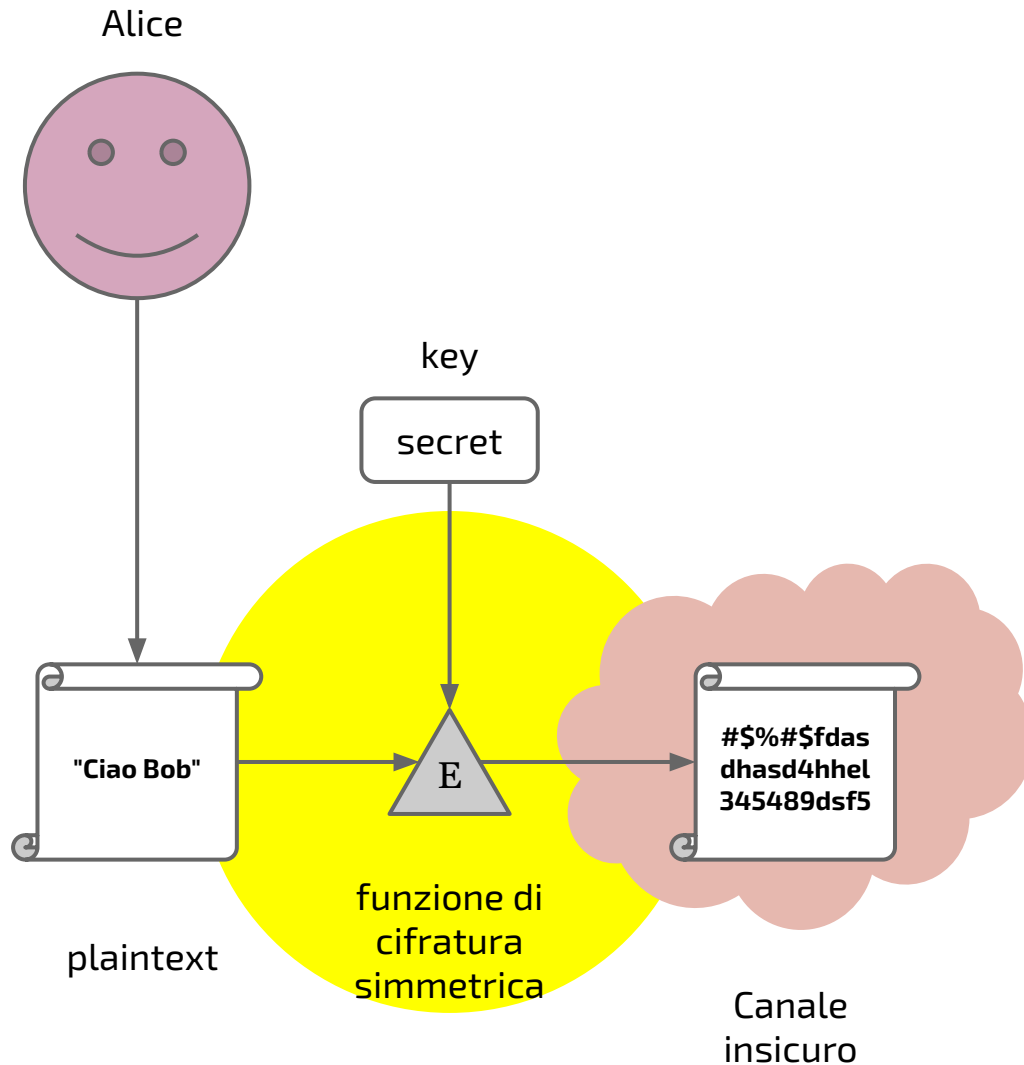
- **Ciphertext attack:** l'attaccante ha solamente messaggi cifrati.
- **Known plaintext attack:** l'attaccante ha delle coppie casuali di messaggi in chiaro e cifrati.
- **Chosen plaintext attack:** l'attaccante può chiedere a un oracolo di cifrargli dei testi per lui.

Punti chiave da ricordare

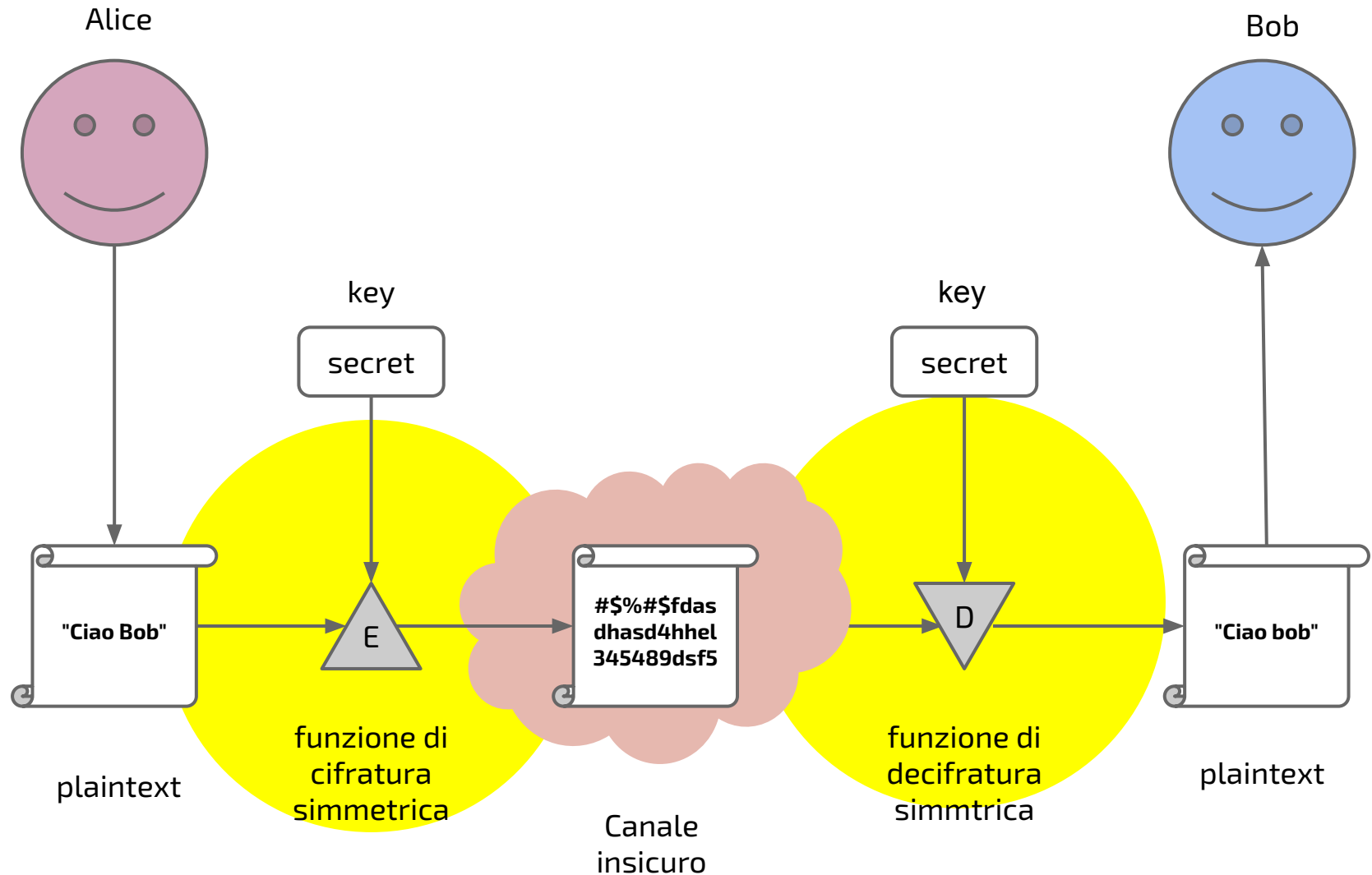
- La sicurezza di un crittosistema è basata sulla robustezza dell'algoritmo.
- Nessun algoritmo, salvo l'OTP, è invulnerabile (ma può richiedere centinaia di anni e condizioni specifiche).
- Un algoritmo è rotto se c'è un attacco più veloce del bruteforcing.
- Non c'è modo di provare la robustezza di un cifrario. Si può solo provare a violarlo.
- La sicurezza di un sistema crittografico deve basarsi solo sulla segretezza della chiave, e mai sulla segretezza dell'algoritmo.
 - I cifrari segreti non sono sicuri. La sicurezza è trasparenza.

Cifratura Simmetrica

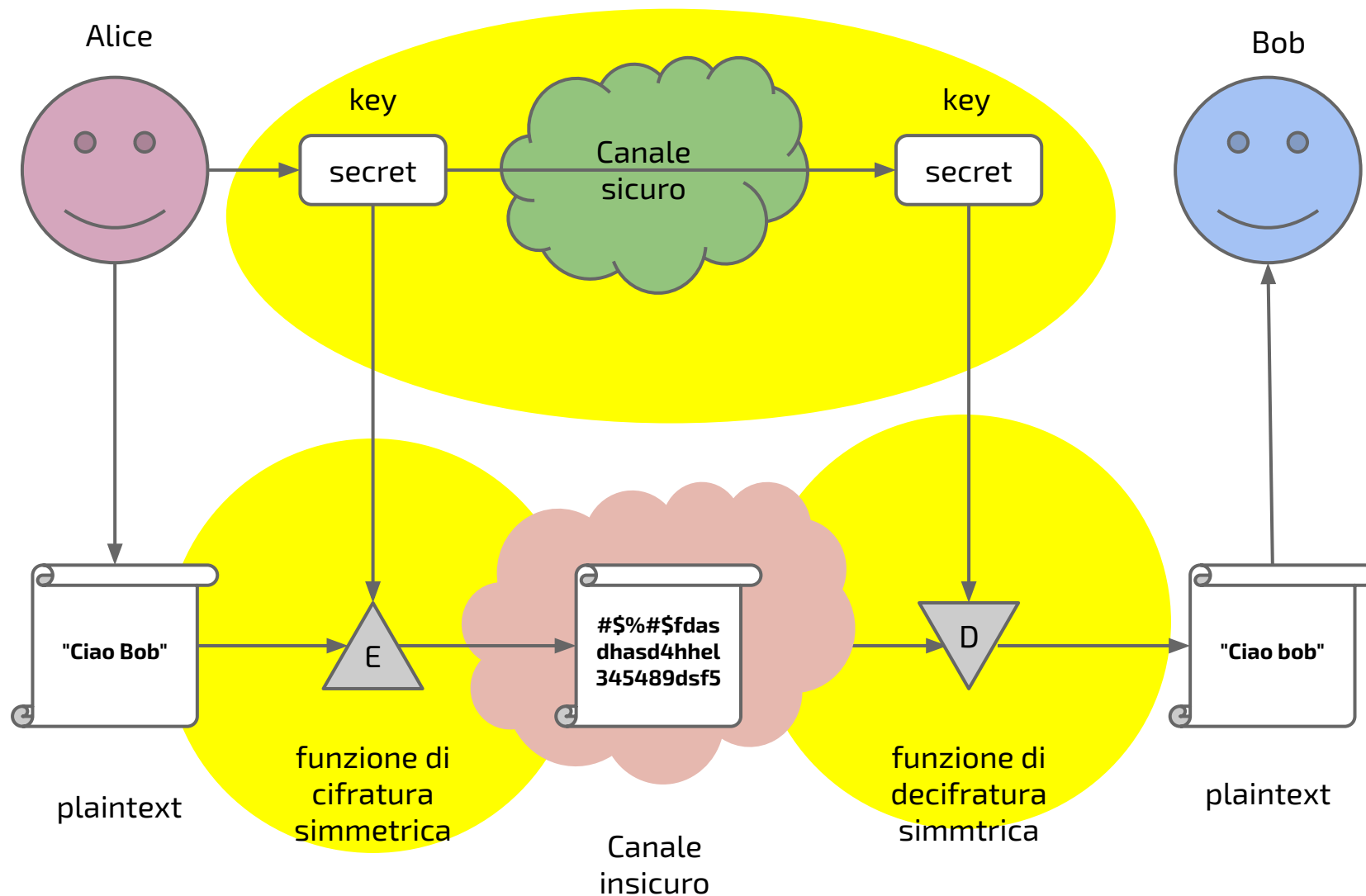
Cifratura simmetrica



Cifratura simmetrica



Cifratura simmetrica




Cifratura simmetrica

- L'idea di base è:
 - Usare la chiave k per cifrare il messaggio.
 - Usare la stessa chiave k per decifrare il messaggio.
- Problema: come scegliamo la chiave?
 - Non possiamo mandare la chiave su un canale non sicuro.
 - Serve un meccanismo di trasmissione su un canale sicuro.

Principi della cifratura simmetrica

- **Diffusione:** distribuire l'informazione del messaggio originale in modo che ogni bit del messaggio cifrato dipenda da molti bit del testo in chiaro.
 - In pratica, una piccola modifica nel messaggio iniziale deve produrre un grande cambiamento nel risultato cifrato.
- 👉 **Obiettivo:** nascondere le relazioni statistiche tra testo in chiaro e testo cifrato.

Principi della cifratura simmetrica

- **Confusione:** La confusione serve a rendere complessa la relazione tra la chiave segreta e il testo cifrato.
 - L'idea è che anche conoscendo molti testi cifrati e i rispettivi testi in chiaro, sia difficile dedurre la chiave.
-  **Obiettivo:** offuscare la dipendenza diretta dalla chiave.

Cifrari simmetrici moderni

- Cifrari moderni mischiano confusione e diffusione
- Tra i più noti:
 - DES (Data Encryption Standard, 1977), e la sua evoluzione 3DES (deprecato)
 - IDEA (1991) (deprecato)
 - BlowFish (1993) (deprecato)
 - RC5 (1994) (deprecato)
 - CAST-128 (1997) (deprecato)
 - **Rijndael** (dal 2000 è chiamato **AES, Advanced Encryption Standard**)
 - **ChaCha** (**ChaCha20**, 2008 circa)

Spazio delle chiavi e brute forcing

Lo spazio delle chiavi in genere è misurato in bit:

- Il tempo di attacco cresce esponenzialmente con il numero di bit (per es., 33 bit richiedono il doppio del tempo rispetto a 32 bit).
- Occorre bilanciare la potenza di calcolo disponibile con la lunghezza della chiave.

Lunghezza chiave (bit)	Tempo di brute force
32	pochi secondi
64	4 mesi
128	5.4×10^{15} anni
256	1.83×10^{54} anni
512	2.1×10^{131} anni

Cifratura *Asimmetrica*

Cifratura Asimmetrica

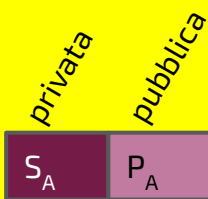
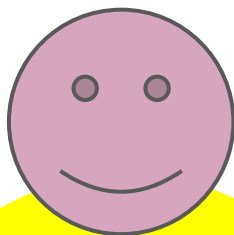
- L'idea di base è:
 - Usare una chiave k_{pu} per cifrare il messaggio.
 - Usare una chiave k_{pr} per decifrare il messaggio.
 - k_{pu} è correlata a k_{pr} e non è possibile ottenere k_{pr} conoscendo k_{pu} , ma l'opposto è necessario.
- Introdotta nel 1976 da W. Diffie & M. Hellman.
- Chiamata anche cifratura a chiave pubblica.
 - Perché la chiave k_{pr} è tenuta **privata**, mentre k_{pu} può essere conosciuta **pubblicamente**.
 - Questo risolve il problema dello scambio di chiavi (quasi!!!).

Cifratura Asimmetrica

- Non descrivere la matematica dietro la cifratura asimmetrica, ma l'idea generale è avere un problema molto complesso dietro la generazione delle chiavi.
 - Fattorizzare un numero: quali sono i divisori di 12847248839274?
 - Logaritmo discreto: $3849532347 = 23^x$. Quanto vale x ?
 - Gli algoritmi più performanti ci impiegano anni per risolvere questi problemi sui computer moderni.
 - Gli algoritmi per **computer quantistici** rompono quasi tutti questi tipi di algoritmi (Post-Quantum Cryptography)

Cifratura Asimmetrica: Scambio di chiavi

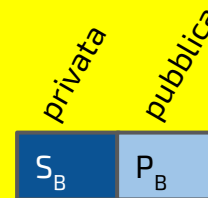
Alice



Coppia di
chiavi

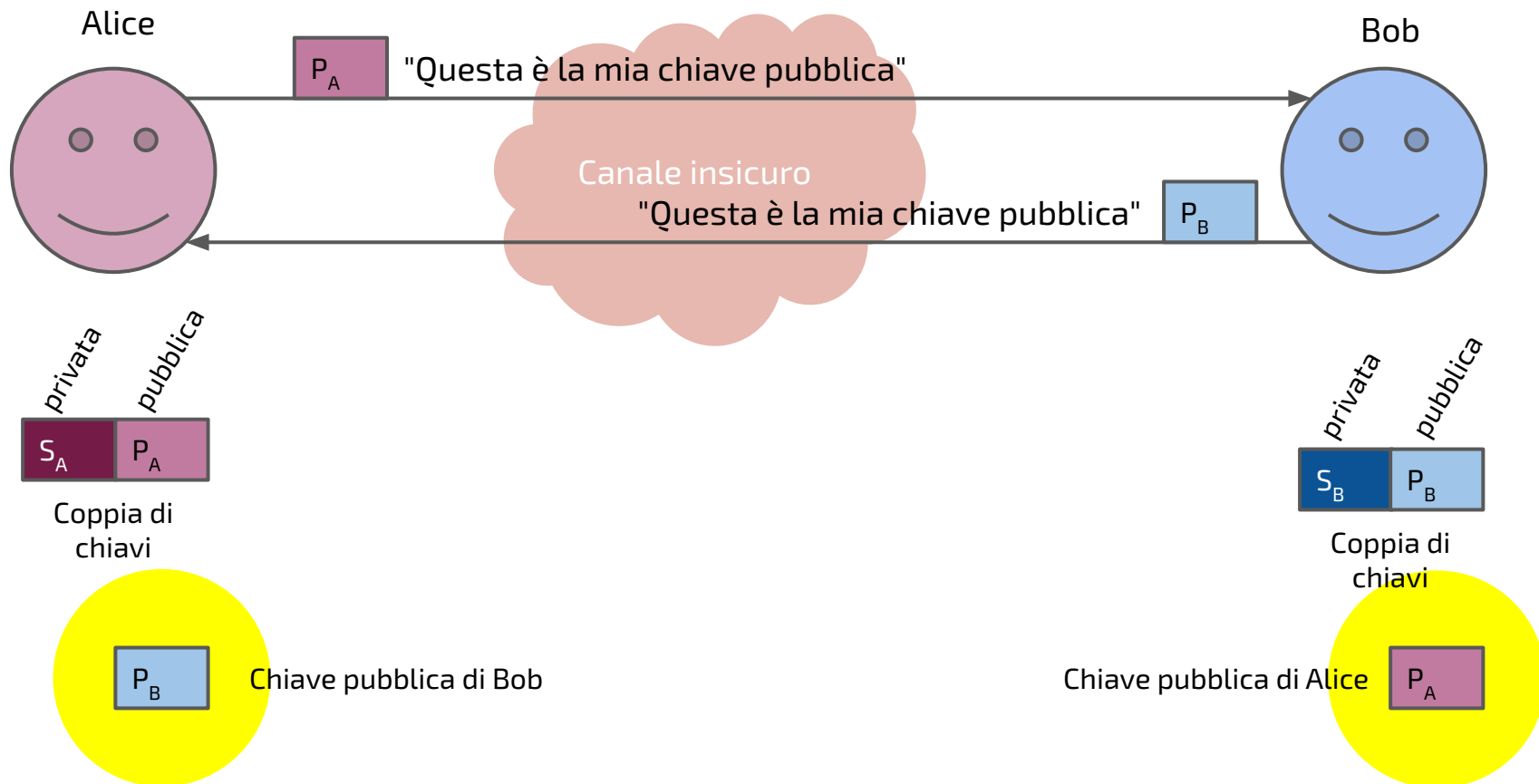
Canale insicuro

Bob

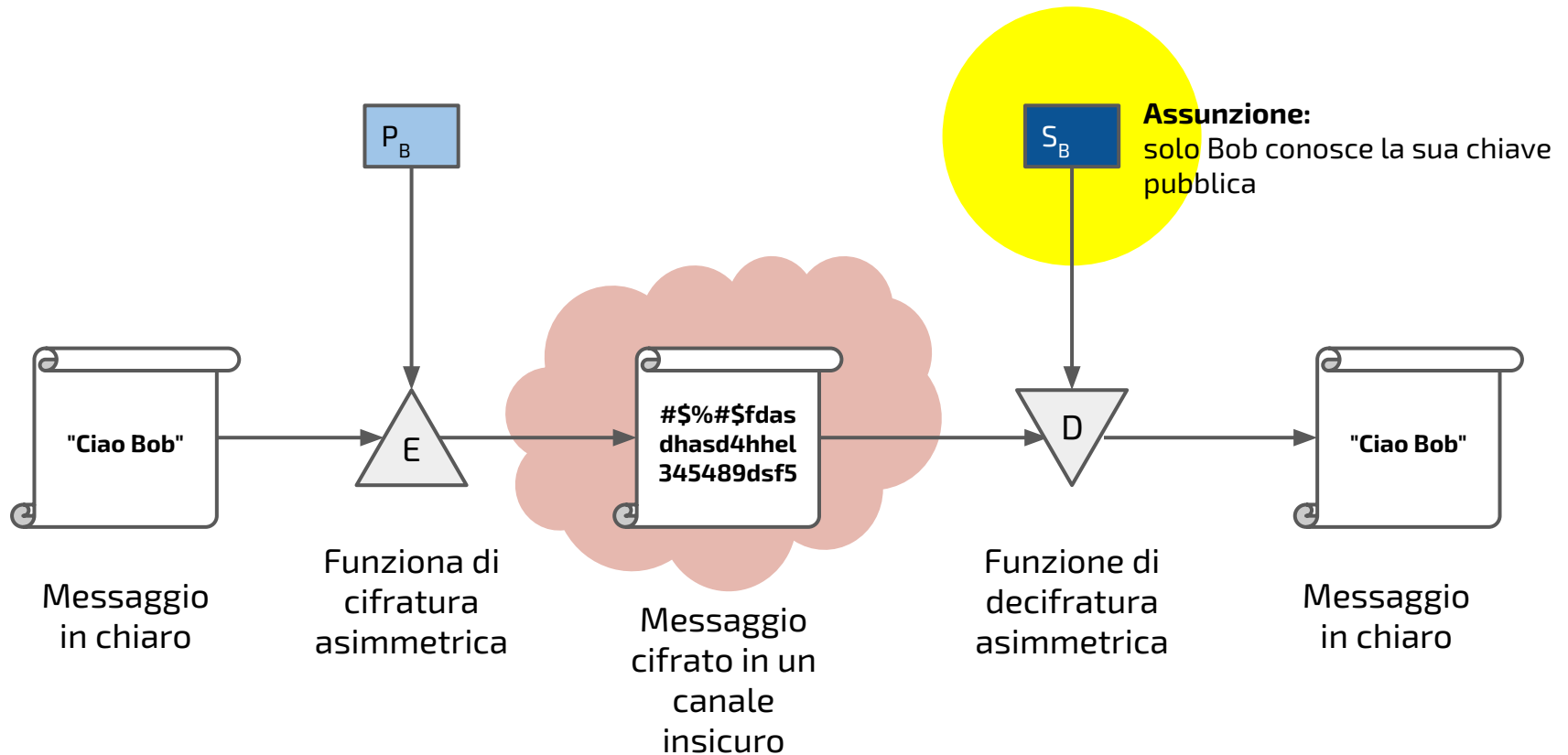


Coppia di
chiavi

Cifratura Asimmetrica: Scambio di chiavi



Cifratura Asimmetrica: Comunicazione

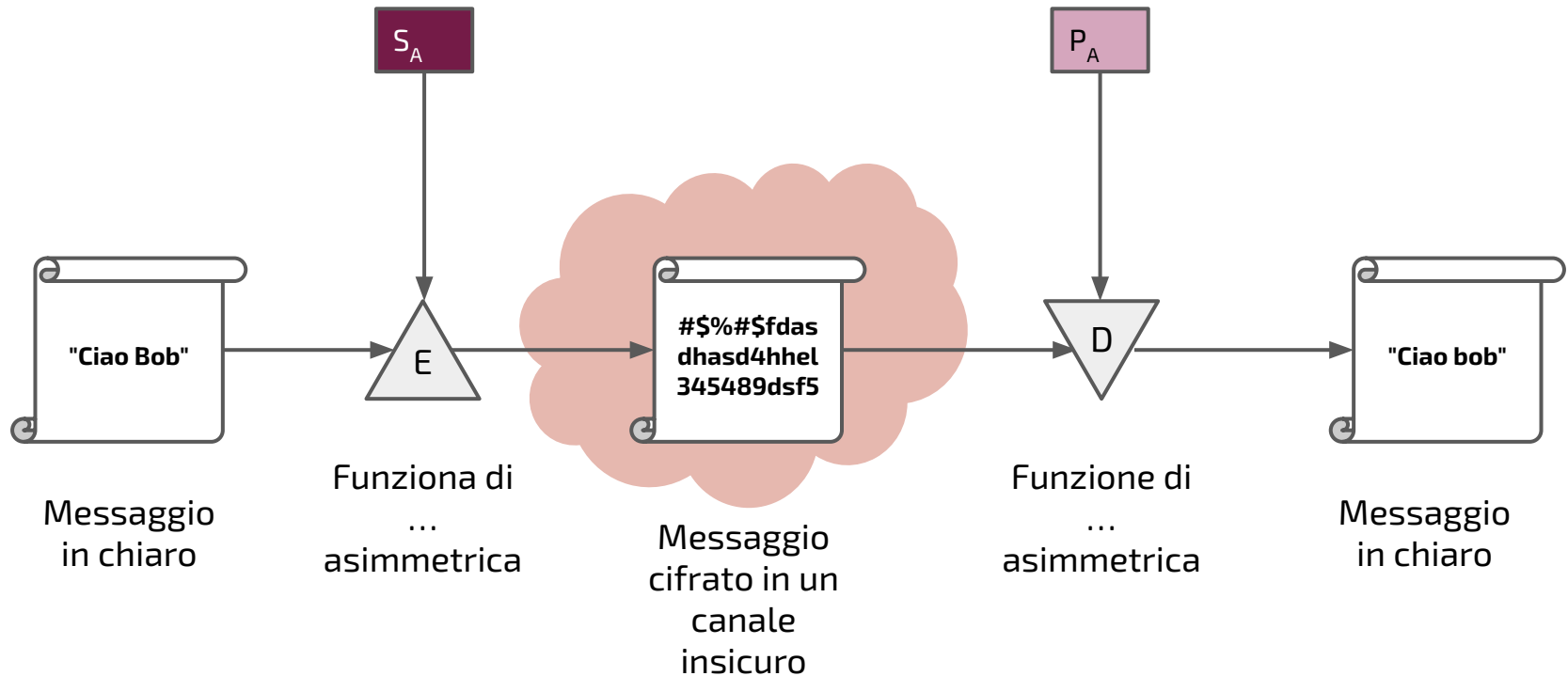


E questo? Funziona lo stesso.

Assumption:

Solo Alice conosce la sua chiave privata

Tutti conoscono la chiave pubblica di Alice

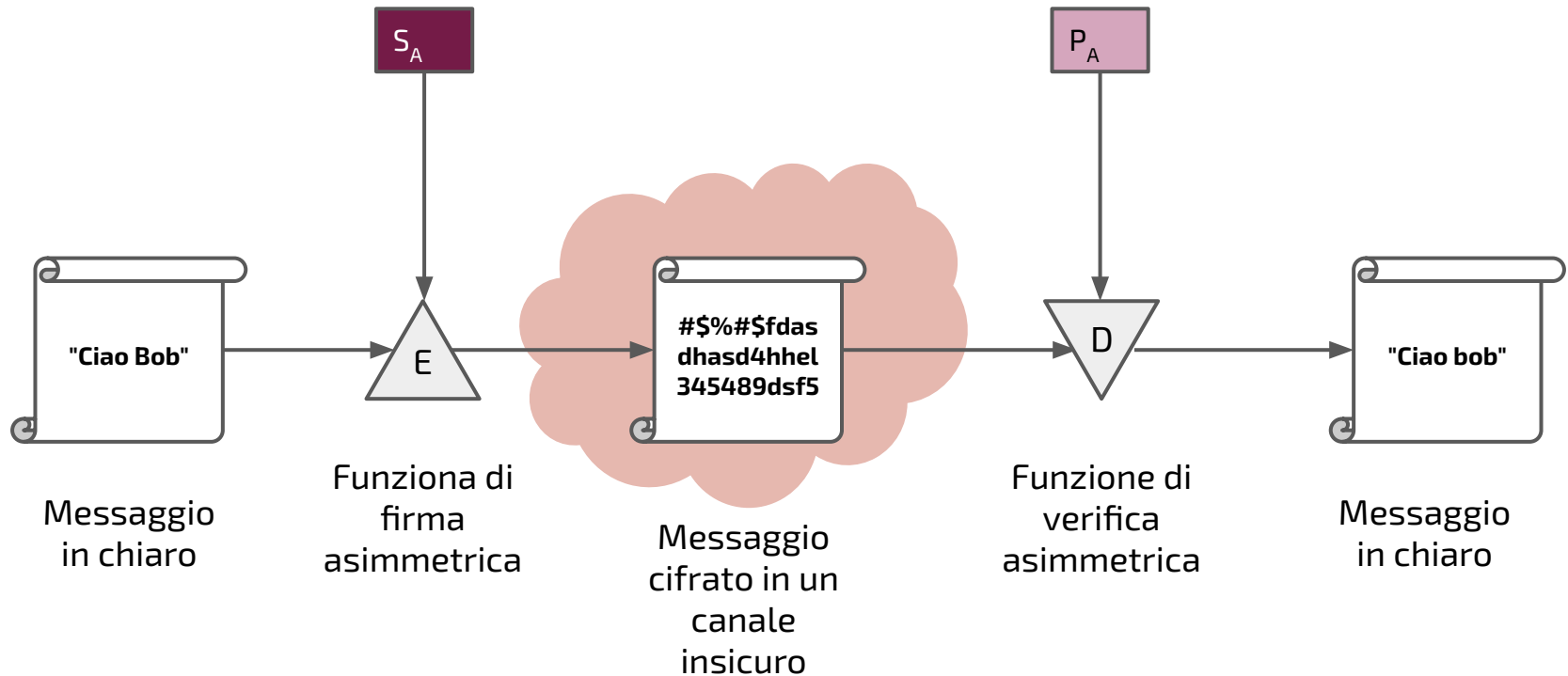


E questo? Funziona lo stesso.

Assumption:

Solo Alice conosce la sua chiave privata

Tutti conoscono la chiave pubblica di Alice



Cifrari Asimmetrici comuni

Una lista dei cifrari asimmetrici più comuni:

- **Diffie-Hellman** (1976)
- **RSA** (1977, Ron Rivest, Adi Shamir, Len Adleman)
- **DSS** (1991, FIPS PUB 186)
- **ECC** (IEEE P1363, elliptic curve cryptography)

L'operatore Modulo

L'operatore modulo è usato molto in crittografia, specialmente nella cifratura asimmetrica. Per noi è semplicemente il resto della divisione:

$$30 \bmod 8 = 6$$

Esempio: scambio Diffie-Hellman

- Usato da Alice e Bob per accordarsi su un segreto attraverso un canale non sicuro.
 - Due persone parlano in mezzo alla classe: tutti li sentono, ma alla fine solo quelle due persone conoscono un segreto, e nessun altro.
- Logaritmo Discreto:
 - Se $y = a^x$ allora $x = \log_a y$
 - Dati x, a, p , è facile calcolare $y = a^x \bmod p$, ma sapendo y , è difficile calcolare x
 - Qui "difficile" significa "computazionalmente molto oneroso": in pratica, il problema richiede una ricerca esaustiva (bruteforce) su tutti i possibili valori di x .

Esempio: scambio Diffie-Hellman

Scegli p primo e a radice primitiva di p , valori pubblici.

Radice primitiva: un numero a tale che, elevandolo a qualsiasi potenza compresa tra 1 e $(p - 1) \bmod p$, si ottiene ogni numero compreso tra 1 e $(p - 1)$

- Esempio: 3 è una radice primitiva di 7 perché
 - $3^1 \bmod 7 = 3$ $3^2 \bmod 7 = 2$ $3^3 \bmod 7 = 6$
 - $3^4 \bmod 7 = 4$ $3^5 \bmod 7 = 5$ $3^6 \bmod 7 = 1$

Quindi, poniamo $a = 3$, $p = 7$ noti a tutti nel sistema.

Esempio: scambio Diffie-Hellman



Chiave privata (segreta):

Loro scelgono un numero X tra $[1, 2, \dots, (p-1)]$

	Alice	X_A	$X_A = 3$
	Bob	X_B	$X_B = 1$

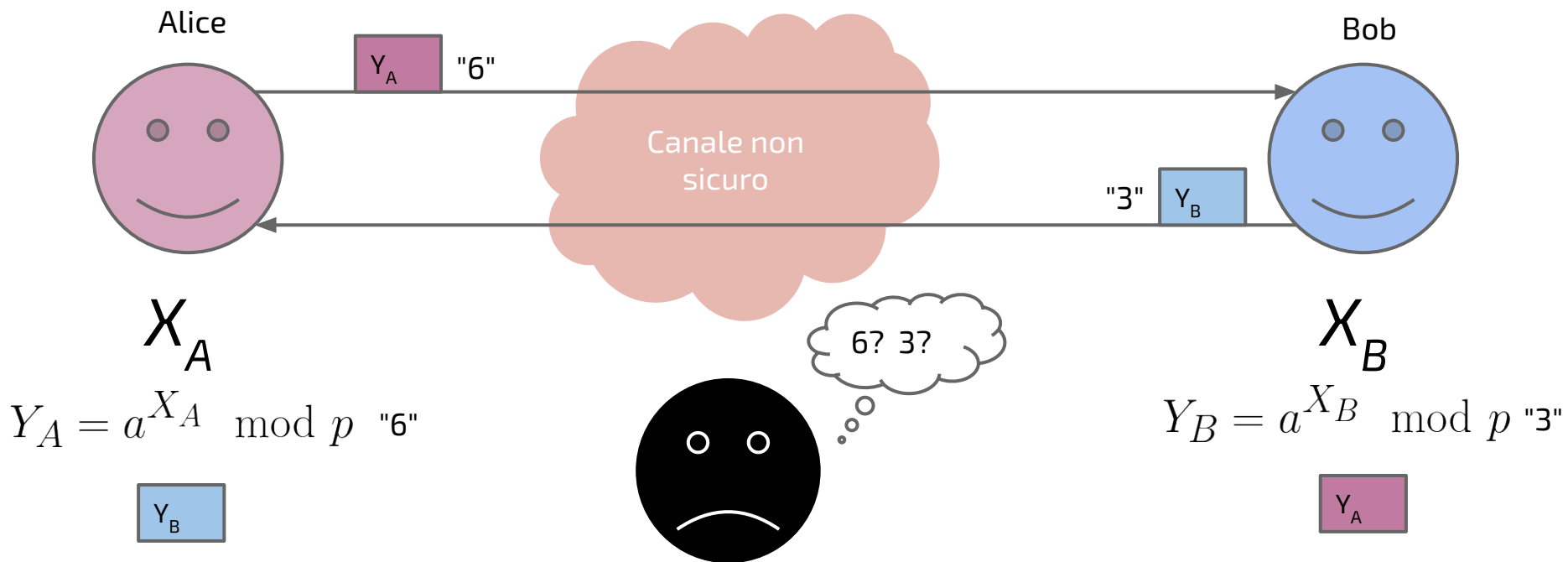
Chiave privata (nota a tutti):

Loro ottengono la loro chiave privata facendo:

	Alice	$Y_A = 3^3 \bmod 7 = 27 \bmod 7 = 6$
	Bob	$Y_B = 3^1 \bmod 7 = 3$

Esempio: scambio Diffie-Hellman

$$a = 3, p = 7$$



Esempio: scambio Diffie-Hellman

A questo punto, loro possono calcolare un **segreto** comune K

- Visto che
$$\begin{aligned} Y_A^{X_B} \bmod p &= (a^{X_A} \bmod p)^{X_B} \bmod p = (a^{X_A})^{X_B} \bmod p = \\ (a^{X_B})^{X_A} \bmod p &= (a^{X_B} \bmod p)^{X_A} \bmod p = Y_B^{X_A} \bmod p \end{aligned}$$
- Alice $K_A = Y_B^{X_A} \bmod p = 3^3 \bmod 7 = 6$
- Bob $K_B = Y_A^{X_B} \bmod p = 6^1 \bmod 7 = 6$

Tutti possono ascoltare, ma nessuno può calcolare la chiave segreta...

- Perché gli manca la chiave privata

Un intuizione sull'algoritmo RSA

Stesso principio, ma il problema è differente (fattorizzazione).

Se p e q due numeri **primi**:

- Calcolare $n = p * q$ è **facile**.
- n è la **chiave pubblica**, p e q la **chiave privata**.
- Ma dato n è estremamente **lento** calcolare p e q ;
 - quadratic sieve field, prova "prova tutti numeri primi fino alla radice quadrata di n ".
- Problema diverso dal logaritmo modulare (Diffie–Hellman), ma si può dimostrare che sono collegati.

Lunghezza delle chiavi: avvertenze

- La lunghezza della chiave si misura in bit, sia negli algoritmi **simmetrici** che in quelli **asimmetrici**.
- Tuttavia, rappresenta concetti diversi nei due casi.
 - Simmetrici: il numero di tentativi di decrittazione necessari (brute force).
 - Asimmetrici: il numero di tentativi necessari per “rompere” la chiave (ossia ricavarla matematicamente).
- Quindi:
 - È possibile confrontare tra loro algoritmi simmetrici in base alla lunghezza della chiave (es. CAST-128 bit è “più debole” di AES-256).
 - Non è possibile confrontare direttamente algoritmi asimmetrici basandosi solo sulla lunghezza della chiave.
 - E, cosa ancora più importante, non confrontare mai direttamente la lunghezza delle chiavi asimmetriche con quelle simmetriche!

Lunghezza delle chiavi: avvertenze

Protection	Symmetric	Factoring Modulus	Discrete Logarithm Key	Discrete Logarithm Group	Elliptic Curve	Hash
Legacy standard level <i>Should not be used in new systems</i>	80	1024	160	1024	160	160
Near term protection <i>Security for at least ten years (2025-2028)</i>	128	3072	256	3072	256	256
Long-term protection <i>Security for thirty to fifty years (2025-2068)</i>	256	15360	512	15360	512	512

<https://www.keylength.com/en/3/>

La prospettiva dei sistemi

“Probabilmente hai visto la porta di un caveau di una banca... dieci pollici di spessore, acciaio temprato, con grossi catenacci... Spesso troviamo l'equivalente digitale di una porta del genere installato in una tenda. Le persone che le stanno attorno discutono su quanto dovrebbe essere spessa la porta, invece di dedicare il loro tempo a guardare la tenda.”

(Niels Ferguson & Bruce Schneier, Practical Cryptography)

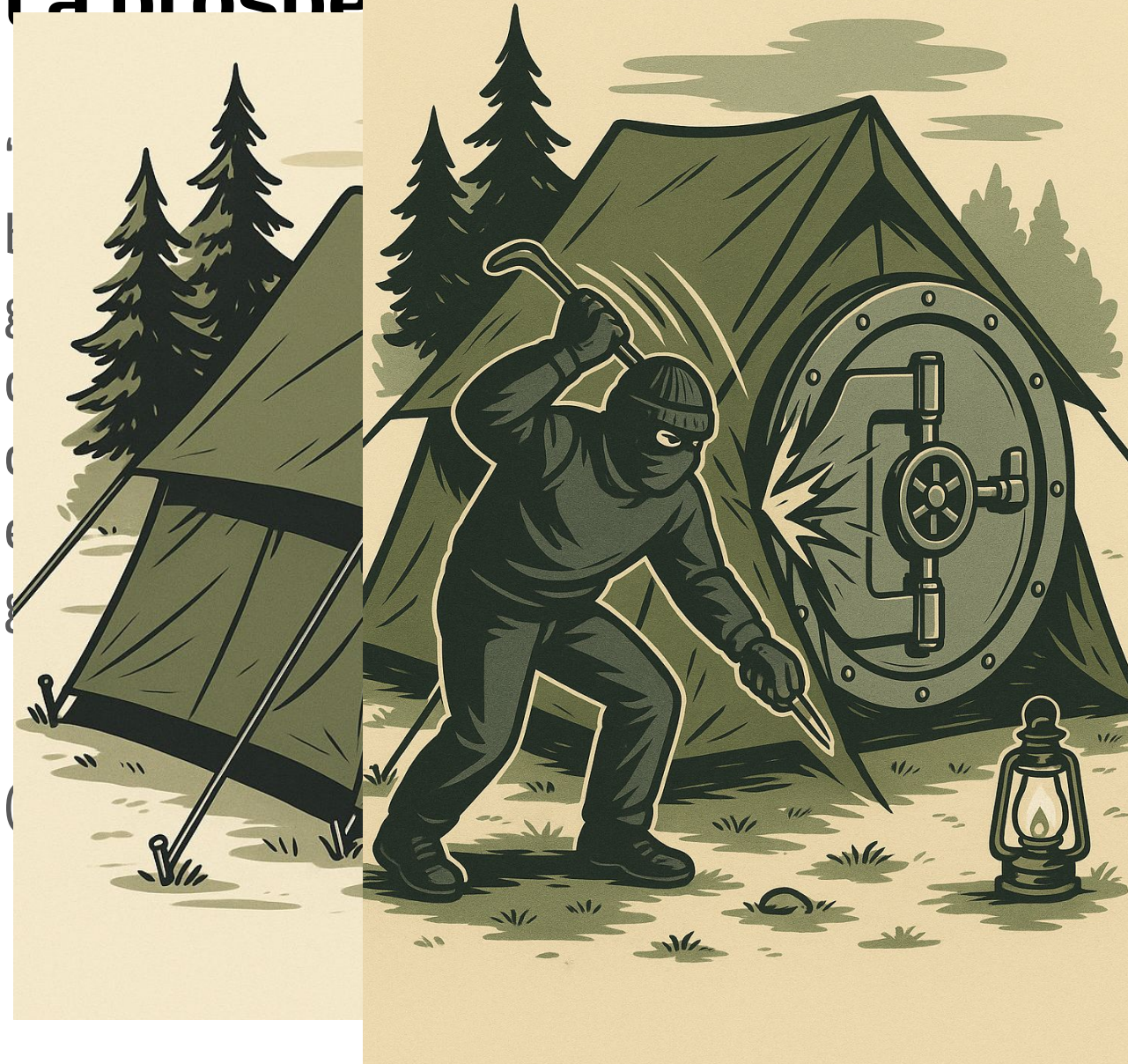
La prospettiva dei sistemi



caveau di una
o temprato, con
quivalente digitale
a tenda. Le persone
anto dovrebbe
are il loro tempo a

tical Cryptography)

La prospe



beau di una
nprato, con
alente digitale
nda. Le persone
dovrebbe
loro tempo a

(Cryptography)

La prospe



ne

y)

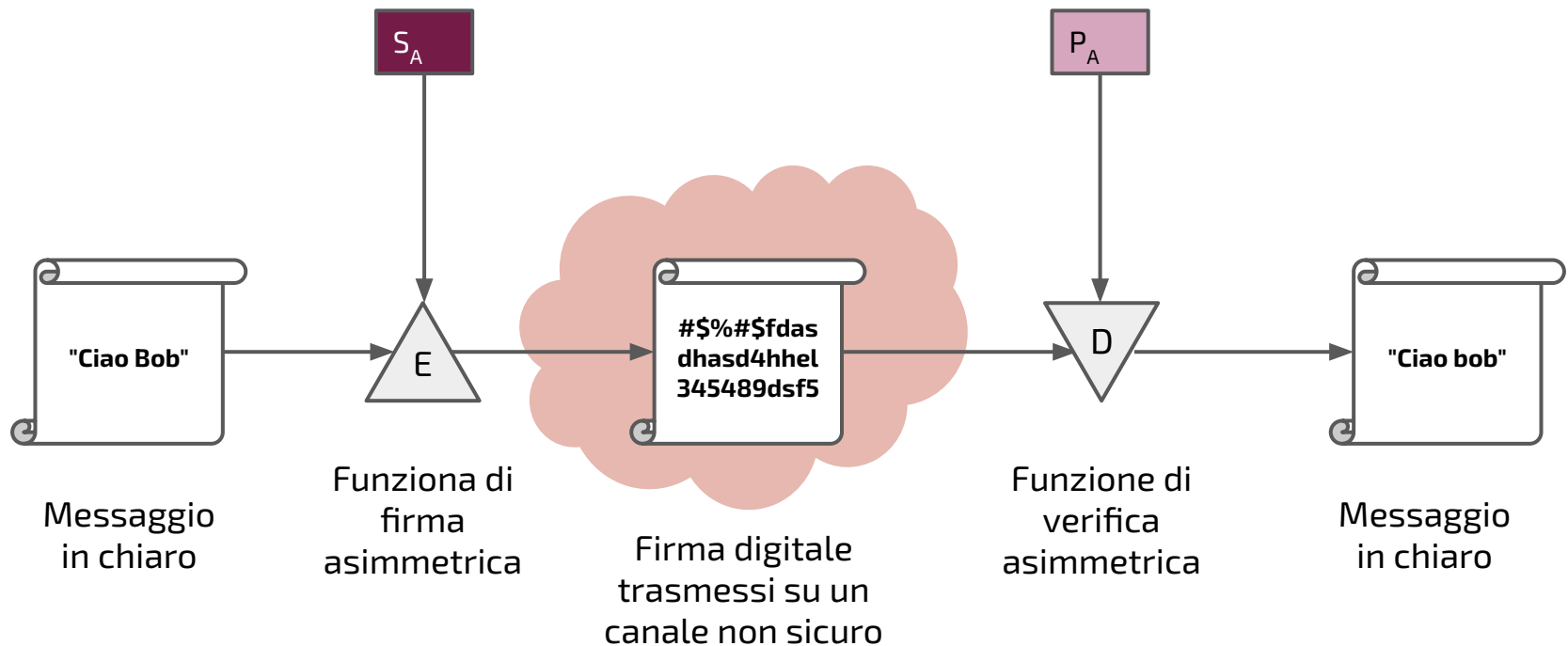
Firma Digitale

Autenticazione del messaggio

Assumption:

Solo Alice conosce la sua chiave privata

Tutti conoscono la chiave pubblica di Alice



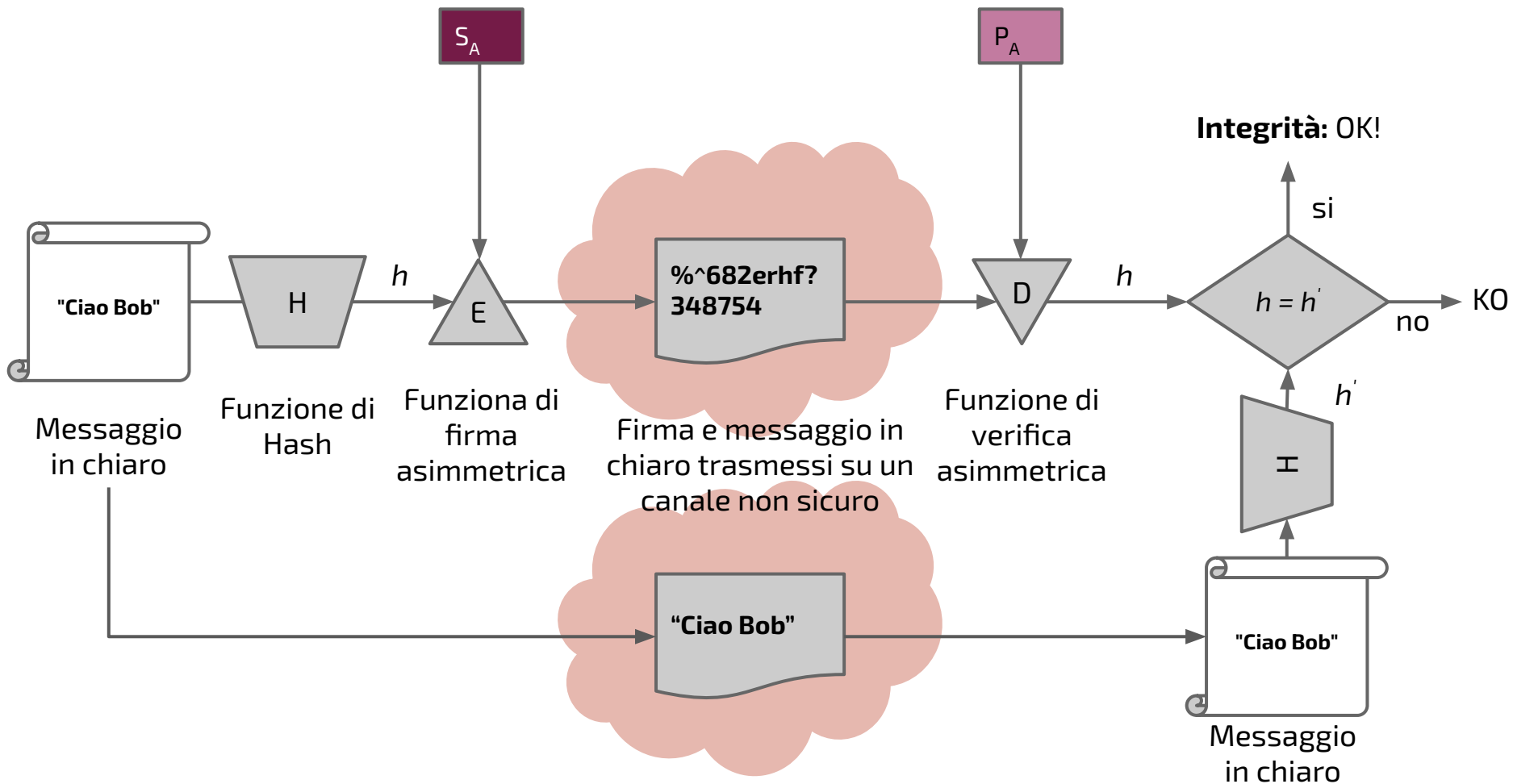
Tuttavia non possiamo firmare digitalmente messaggi di lunghezza arbitraria (limite matematico). Dunque?

Autenticazione del messaggio

Assumption:

Solo Alice conosce la sua chiave privata

Tutti conoscono la chiave pubblica di Alice



Una questione di identità

- Una firma digitale garantisce che il testo in chiaro sia stato redatto da qualcuno.

Una questione di identità

- Una firma digitale garantisce che il testo in chiaro sia stato redatto da qualcuno.
- **Non proprio!** Garantisce solo che il messaggio è stato firmato (o cifrato) con una certa chiave privata... non dice nulla su chi stia effettivamente usando quella chiave privata.
- Lo stesso vale per l'uso della chiave pubblica nella cifratura: anche in quel caso non si ha garanzia sull'identità dell'utente, ma solo sul possesso della chiave.
- Per questo lo scambio delle chiavi pubbliche deve avvenire in modo sicuro — ad esempio tramite un canale separato (out of band) o attraverso altri meccanismi di verifica.

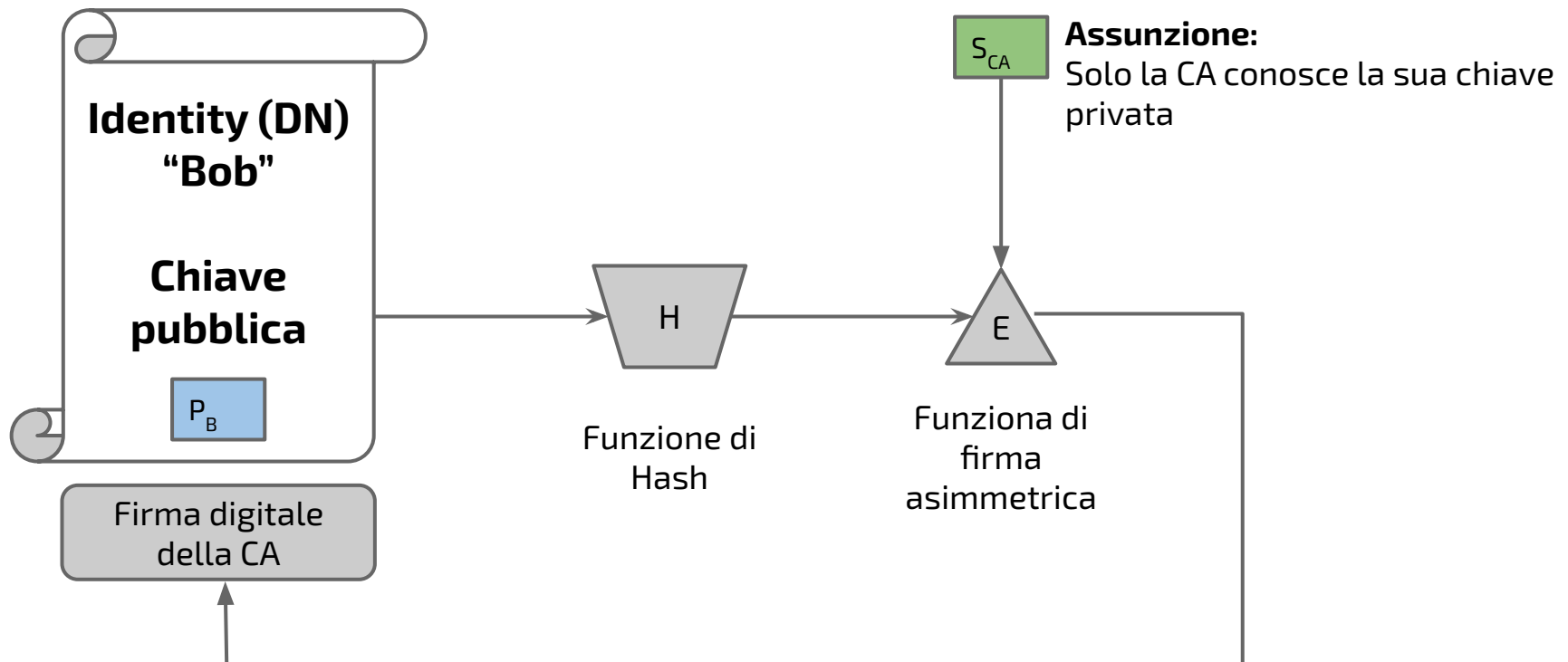
La soluzione non digitale



PKI (Public Key Infrastructure)

- La **PKI (Public Key Infrastructure)** serve proprio a questo: associare le chiavi alle identità in modo affidabile e su larga scala.
- Una PKI (Public Key Infrastructure) utilizza una terza parte fidata, chiamata Autorità di Certificazione (**CA — Certification Authority**).
- La CA firma digitalmente dei file chiamati certificati digitali, che associano un'identità a una chiave pubblica.
- L'identità è rappresentata da un "Distinguished Name (DN)", (l'equivalente di nome e cognome) come definito dallo **standard X.509** (il più utilizzato).
- In questo modo possiamo riconoscere diversi soggetti, a condizione di poter ottenere la chiave pubblica della CA.

Il certificato digitale di Bob



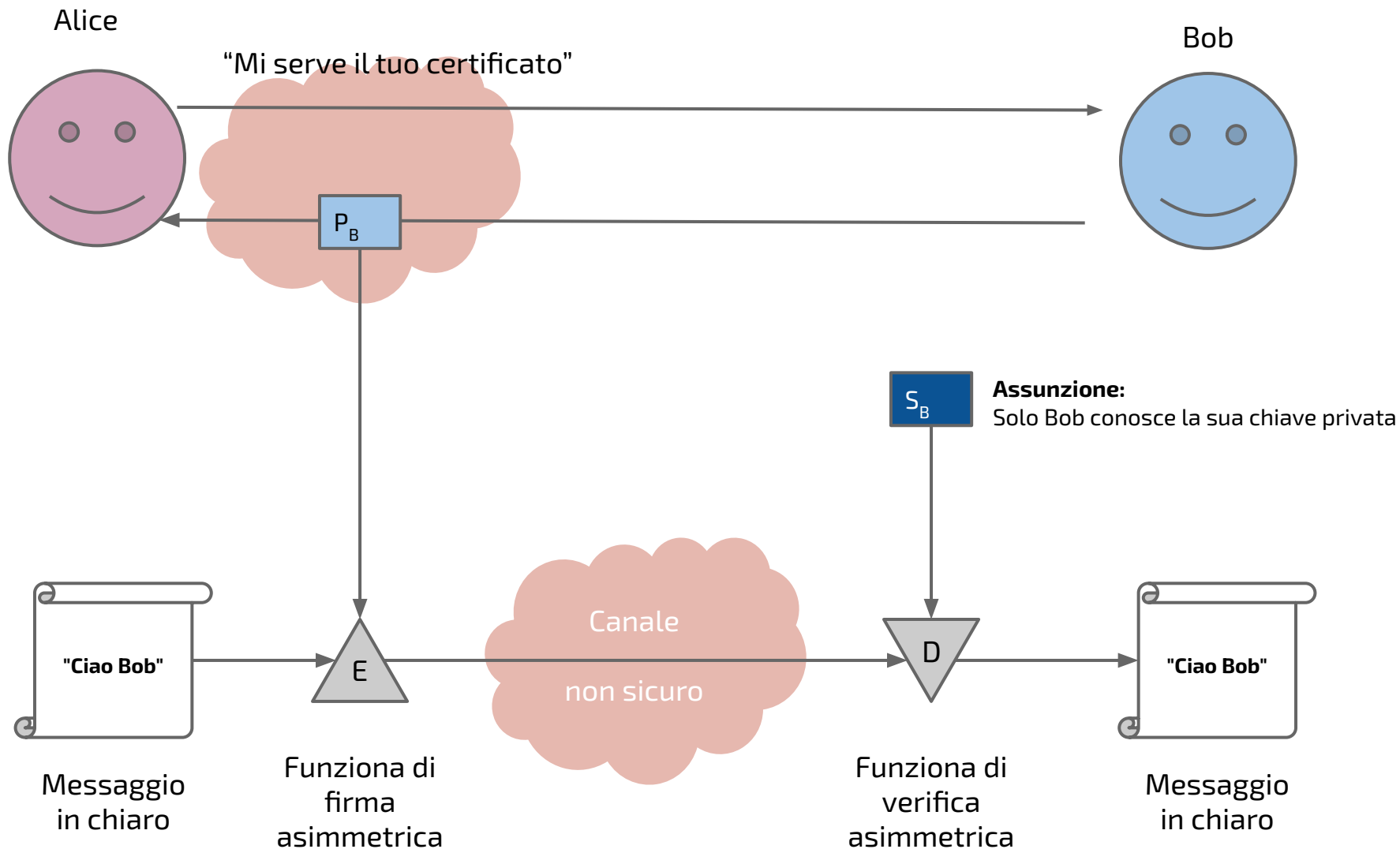
Pro e contro della firma digitale e olografa

Aspetto	Firma olografa	Firma digitale
Supporto	Cartaceo	Elettronico
Autenticità	Basata su grafia	Basata su crittografia
Integrità del documento	Nessuna	Garantita
Facilità d'uso	Molto alta	Media (richiede strumenti)
Valore legale	Pieno su carta	Pieno se qualificata
Rischio di falsificazione	Alto	Basso
Prova di volontà	Forte	Debole

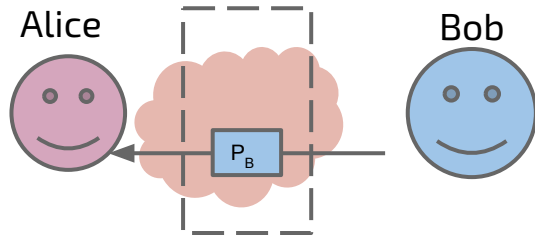
Prova di volontà: a dimostrazione che una persona ha espresso consapevolmente la propria intenzione di firmare un documento e di accettarne il contenuto.

- la chiave privata è stata usata da un altro (abuso, furto, delega impropria);
- la persona ha firmato senza comprendere il contenuto (errore o inganno).

Ottenere il certificato digitale di Bob



Zoom In: è il certificato valido?



Identity (DN)
"Bob"

Chiave
pubblica

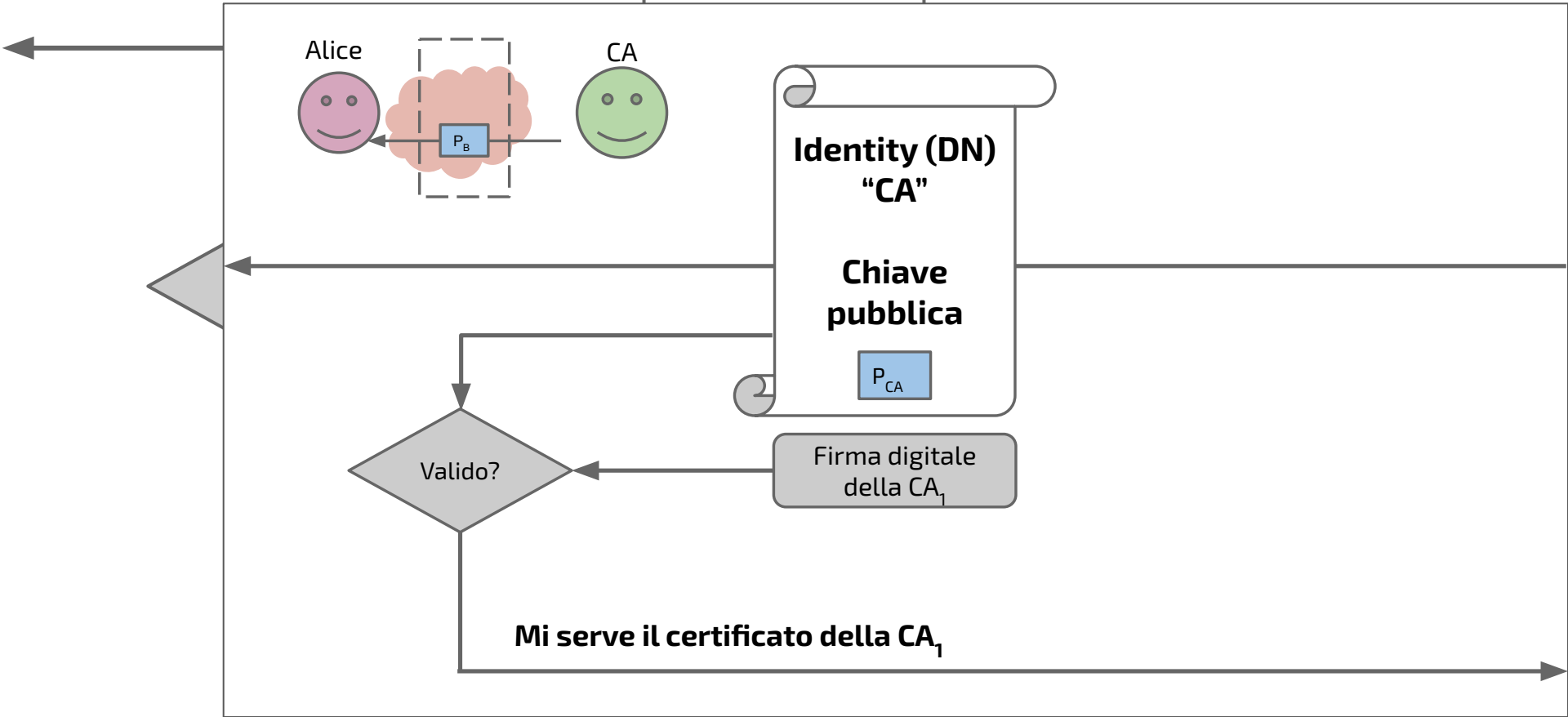
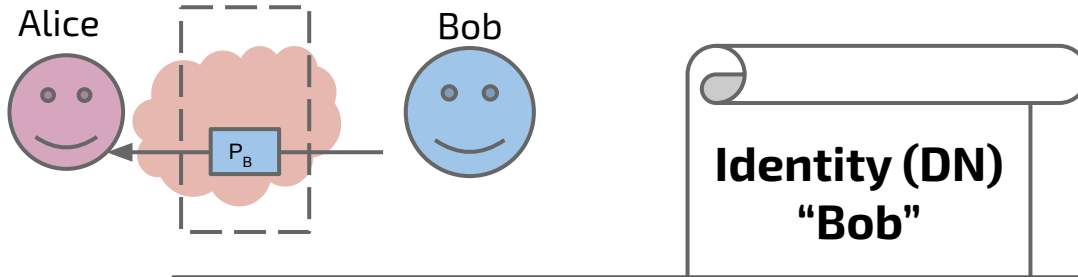
P_B

Firma digitale
della CA

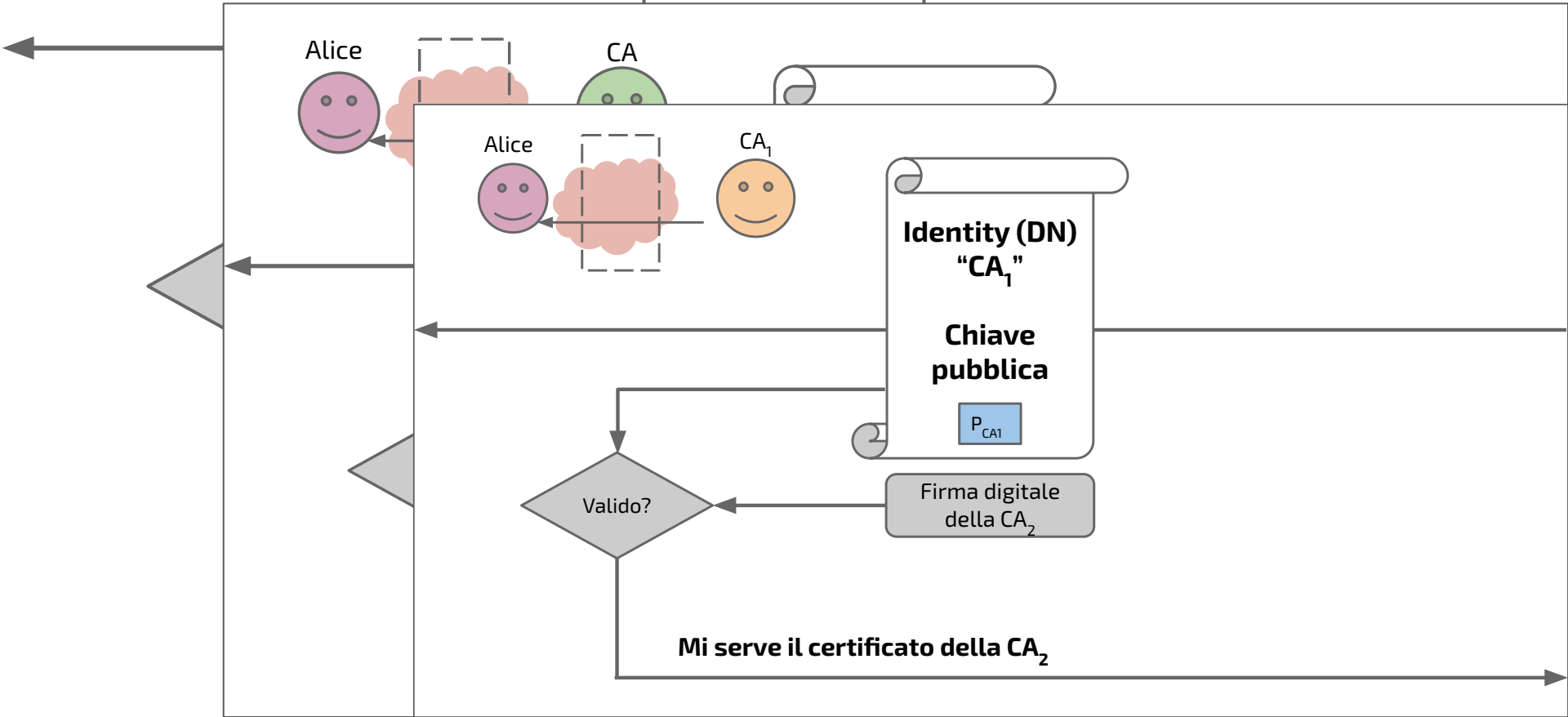
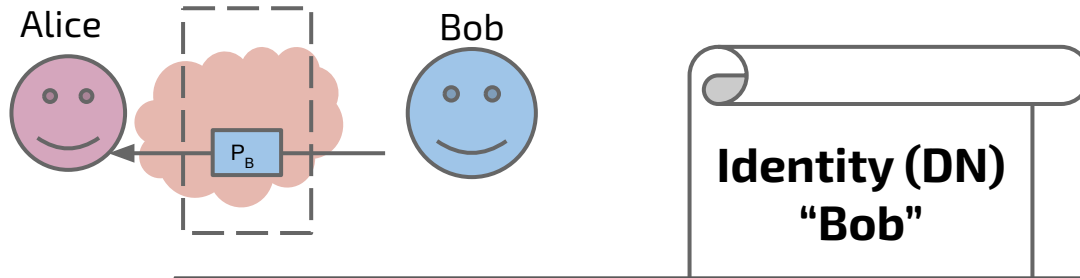
Valido?

Mi serve il certificato della CA

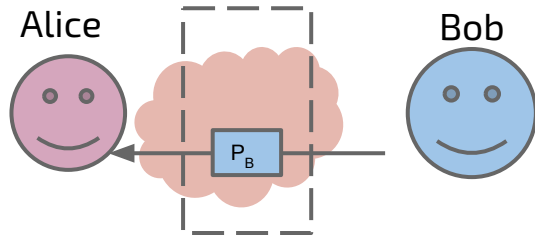
Zoom In: è il certificato valido?



Zoom In: è il certificato valido?



Zoom In: è il certificato valido?



Identity (DN)
"Bob"

Chiave
pubblica

P_B

Firma digitale
della CA

Valido?

Mi serve il certificato della CA

Dove si ferma?

La catena di certificati

La CA ha bisogno di una chiave privata per firmare un certificato.

- La chiave pubblica... deve trovarsi all'interno di un certificato.

Qualcun altro deve quindi firmare quel certificato.

E così via...

- A un certo punto, però, questa catena deve finire

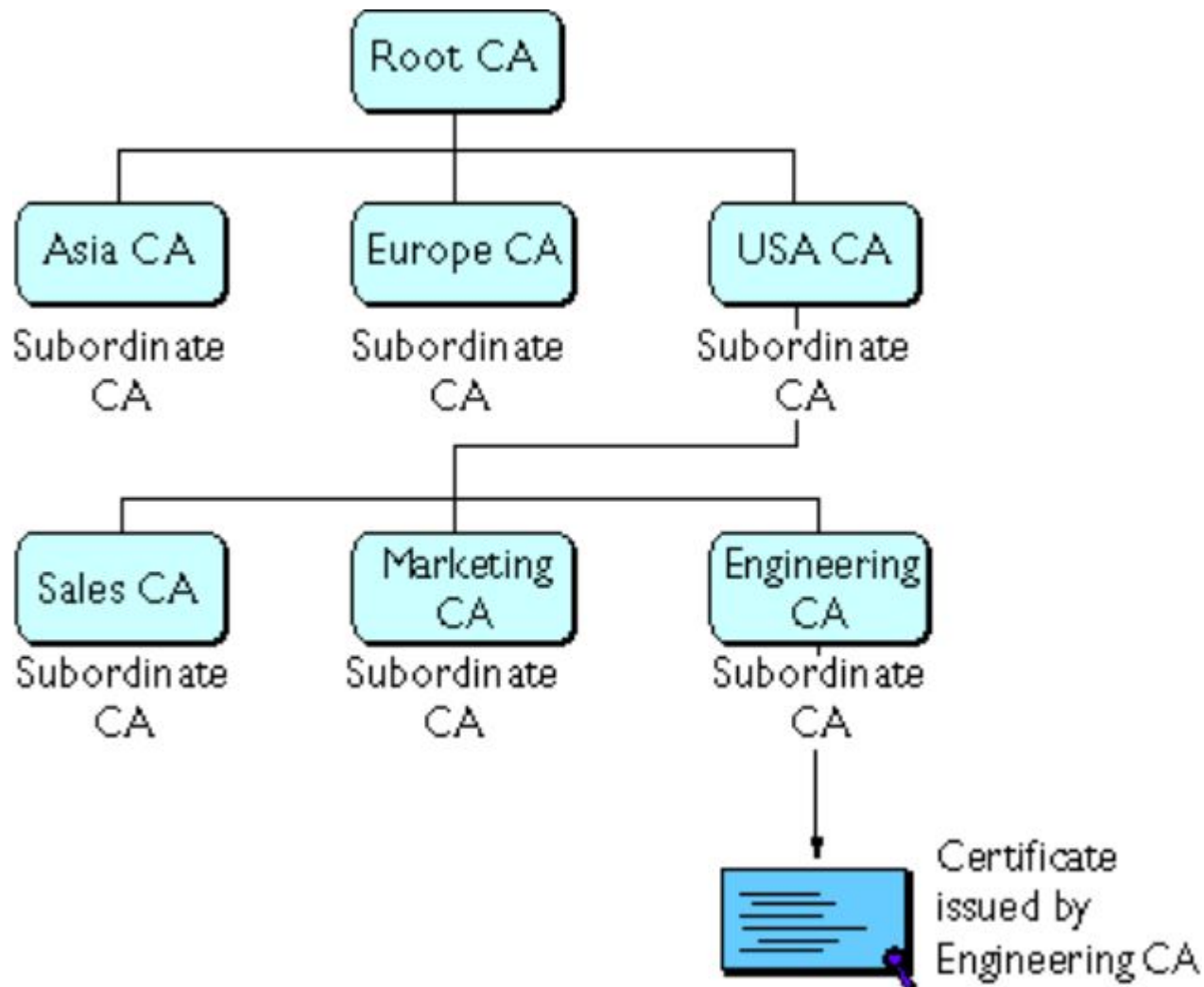
Abbiamo bisogno di un elemento fidato

Top-Level CA (Root CA, Source CA)

Utilizza un certificato auto-firmato.

- In pratica, è un documento che dice: "Io sono me stesso".
- Non può essere verificato: è un elemento considerato fidato.

La catena, o meglio, l'albero



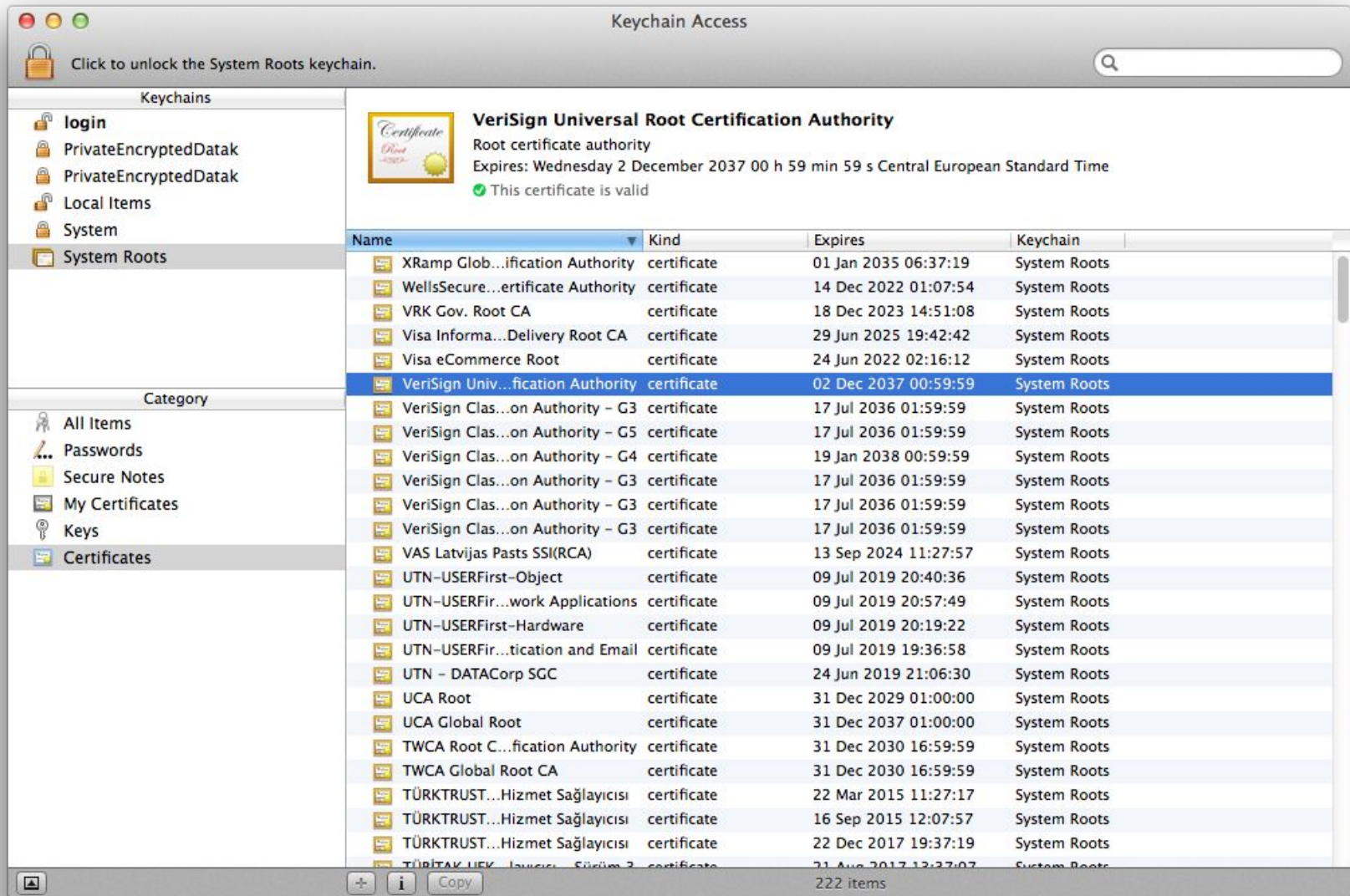
Come distribuire l'elemento fidato?

Un'autorità lo rilascia:

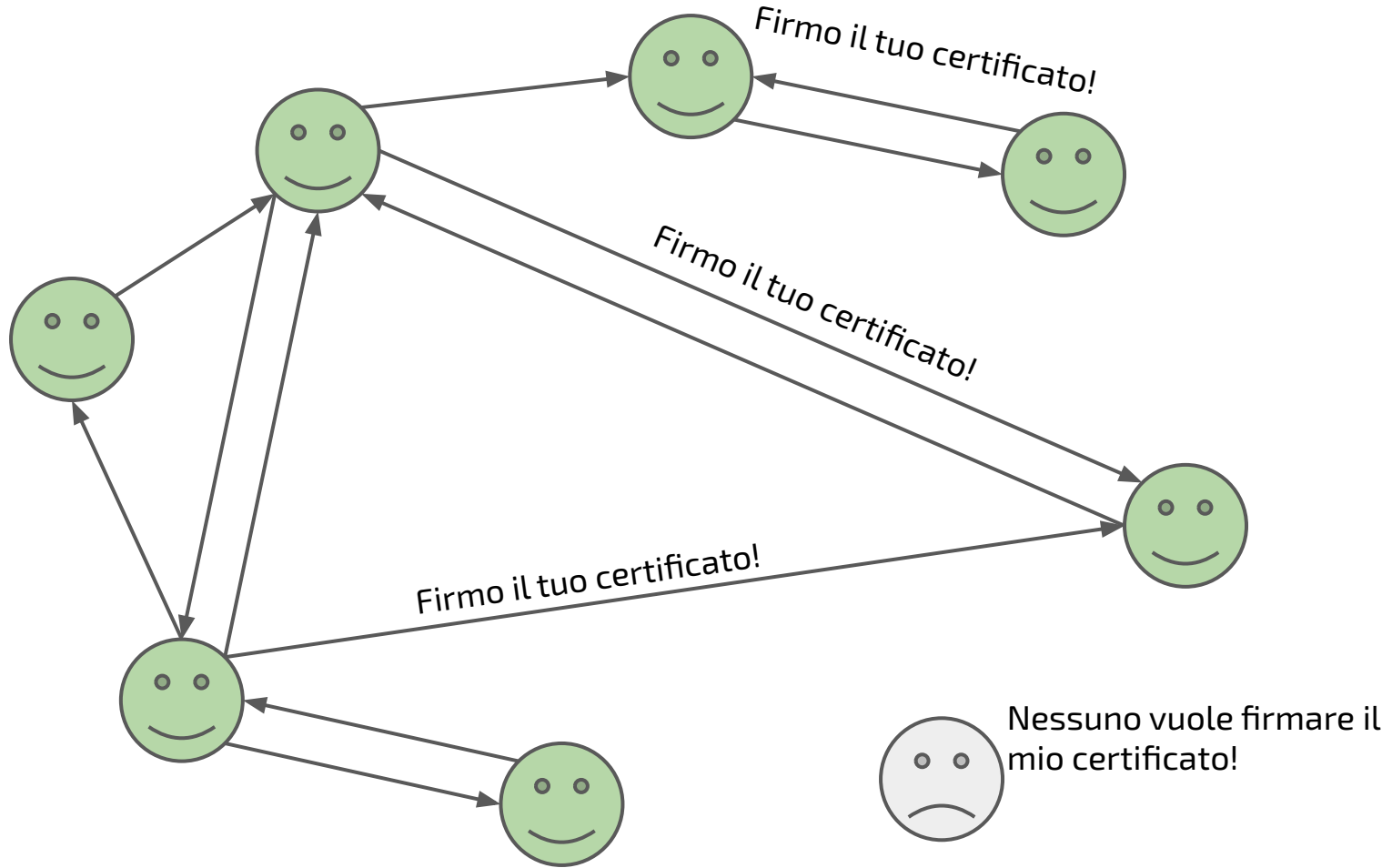
- Lo Stato
- Un ente regolatore
- La direzione dell'organizzazione
- Alcune Big Tech (Google, Microsoft, ...)
- ... e anche le poste!

Alcuni certificati sono già installati nel vostro sistema operativo.

Vi fidate del vostro sistema operativo?



Fiducia decentralizzata: Web of Trust



Problemi di revoca dei certificati

Che succede se la chiave privata associata a un certificato viene resa nota? (**data breach**/violazione dei dati)

- Le firme non possono essere revocate (né distrutte).
- A volte, invece, i certificati devono essere revocati.
- Si utilizzano le **Liste di Revoca dei Certificati (CRL – Certificate Revocation Lists)**.

Sequenza di verifica di una firma

1. La **firma** convalida il documento?
 - Verifica dell'hash, come abbiamo visto.
2. La **chiave pubblica** è quella contenuta nel certificato?
3. Il **certificato** appartiene davvero al soggetto dichiarato?
 - Possibili problemi con soggetti omonimi o con il Distinguished Name (DN).
4. Il **certificato** è validato dalla CA?
 - Bisogna verificare l'intera catena di certificazione, fino alla radice.
5. Il **certificato** top-level radice è fidato?
 - Occorre già possedere il certificato della CA top-level.
6. Questi certificati sono **validi**? Non appartengo alla lista dei certificati revocati?

Ogni controllo che manca è una vulnerabilità

Caso di studio: Quadro normativo italiano sulla firma digitale

Introdotta in Italia con il D.P.R. 513/1997,

- poi modificato più volte, in particolare per adeguarsi ai regolamenti europei (come la direttiva 1999/93/CE e successivamente il regolamento eIDAS n. 910/2014).

Il sistema originario italiano prevedeva una **lista di Autorità di Certificazione (CA) accreditate, controllate e approvate dallo Stato** — quindi una lista “selezionata” o “verificata” di soggetti autorizzati a rilasciare certificati qualificati.

Caso di studio: Quadro normativo italiano sulla firma digitale

Risultato:

Ogni CA ha sviluppato il proprio software di firma digitale, cioè il proprio “trusted element” (elemento fidato, come DiKe, ArubaSign, Actalis, ecc.), con logiche e interfacce diverse ma tutte conformi alle regole tecniche AgID.

Crypto: OK – Software Design: KO

Gli standard italiani di firma utilizzano **algoritmi crittografici forti e non compromessi!**

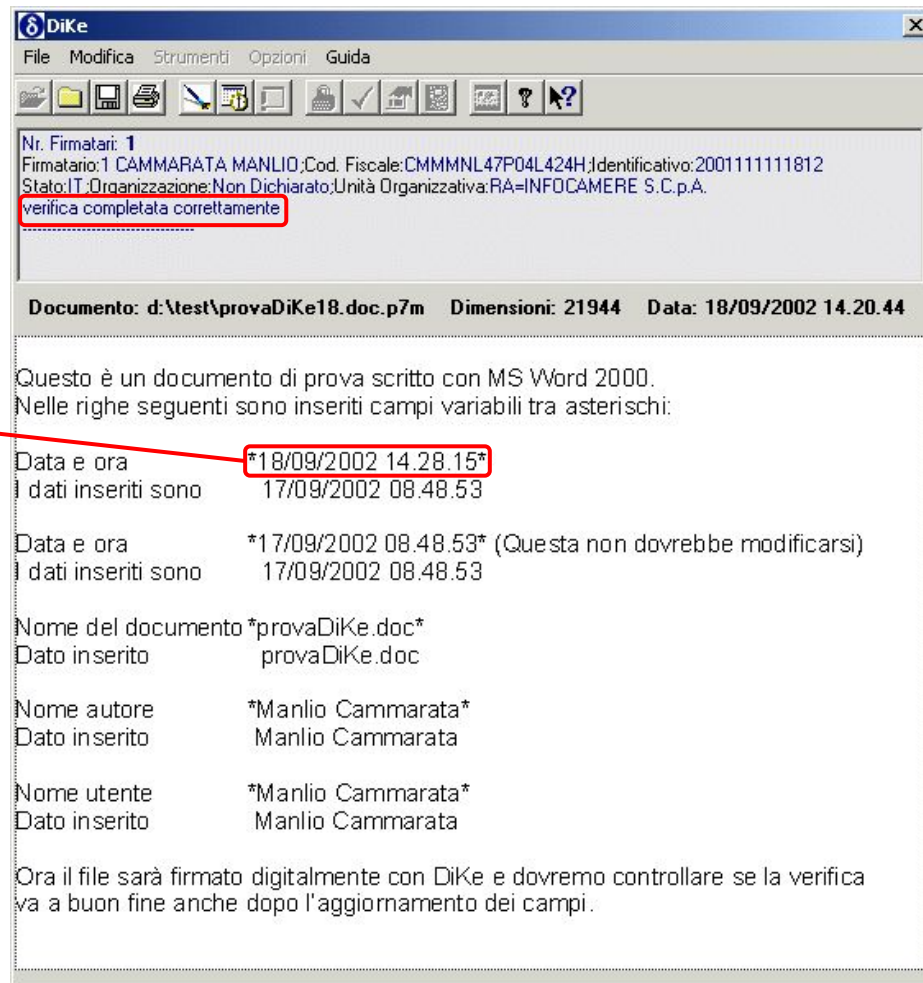
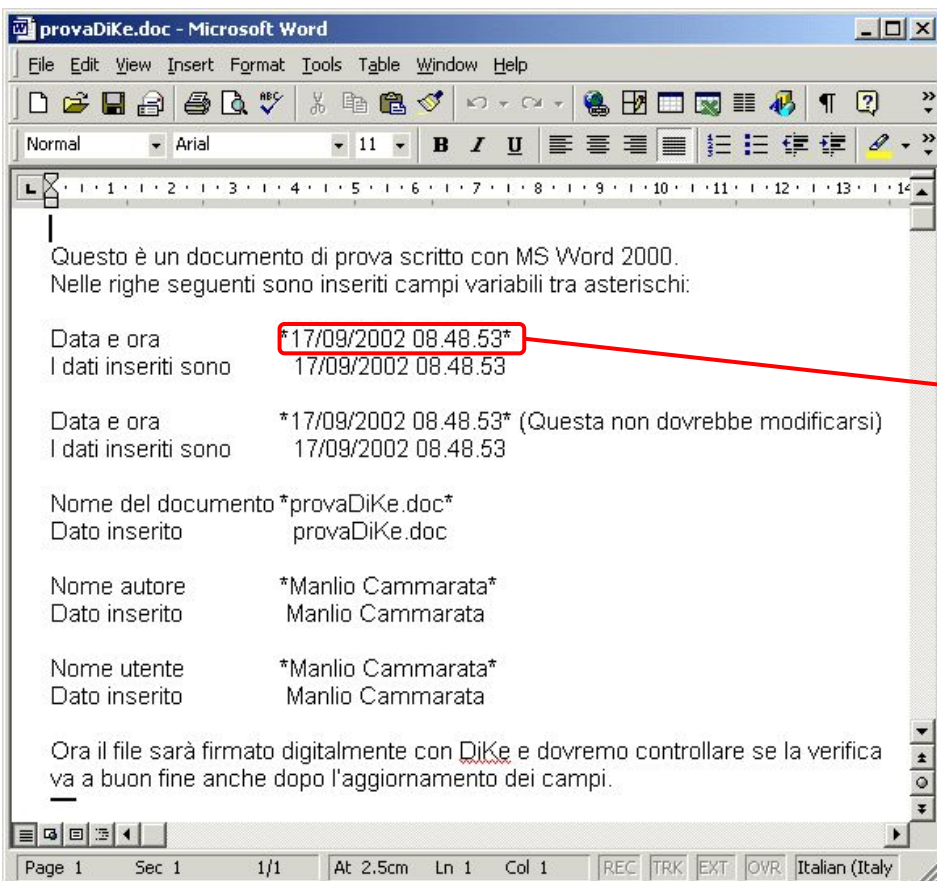
Tuttavia, sono emerse alcune **vulnerabilità**.

Ricordi la “porta del caveau della banca in una tenda”?

Bug 1: Fields of pain

- Bug segnalato il 9 settembre 2002.
- Il software di diverse CA (inizialmente DiKe di Infocamere, oggetto dell'analisi) permetteva agli utenti di firmare documenti Word contenenti campi dinamici o macro senza alcun avviso.
- Una macro non modifica la sequenza di bit del documento, quindi la firma (hash) non cambia anche se il contenuto visualizzato è diverso.


Esempio innocuo



Il problema?

- Si poteva creare una macro che cambiava un prezzo in base alla data.
 - Oggi il prezzo è 1000, domani è 5000.
- Vedete un problema?

Reazione

- Le CA risposero che si trattava di un “comportamento previsto” e che non violava la legge.
- Tuttavia:
 - Il 30 gennaio 2003, Microsoft rilasciò una patch per Office che permetteva di disabilitare le macro tramite API.
 - Oggi, tutti i software mostrano un grande avviso quando si tenta di firmare un documento Office.
 - Le nuove normative escludono esplicitamente i formati modificabili o contenenti script (ma raccomandano l'uso del PDF).
- In realtà, la questione è molto più complessa!
- Questo ha dato origine al campo di ricerca noto come  “What You See Is Not What You Sign” (Ciò che vedi non è ciò che firmi).

Bug2: Firma&Cifra

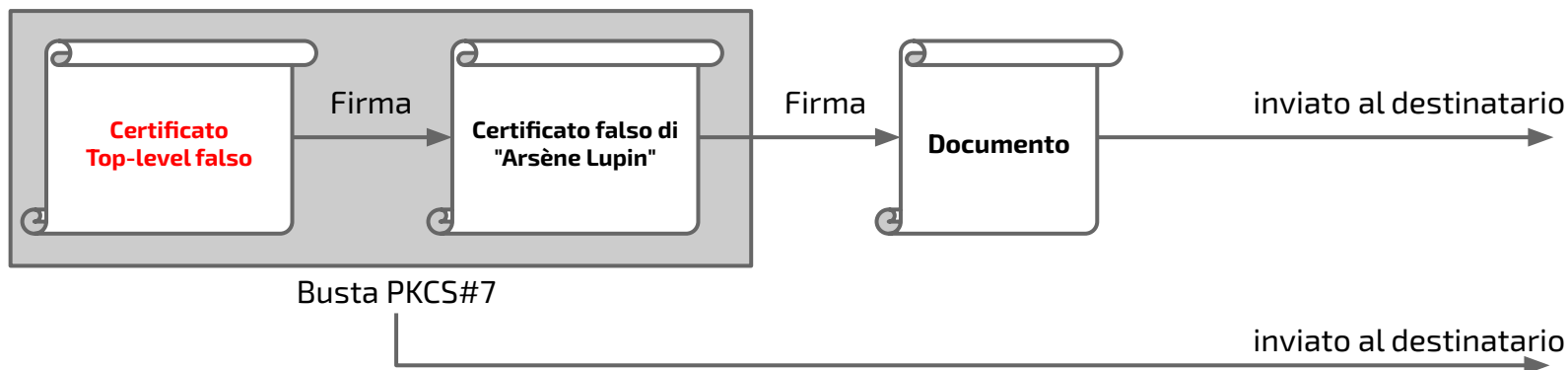
- Firma&Cifra era l'applicazione di firma digitale di PosteCom.
- Un bug è stato scoperto da un anonimo il 20 marzo 2003:
<http://www.interlex.it/docdigit/sikur159.htm>
- **Risultato:** era possibile creare e verificare una firma utilizzando un certificato falso.
- Anche in questo caso, nessun algoritmo crittografico è stato violato per dimostrare il problema.

Descrizione della vulnerabilità

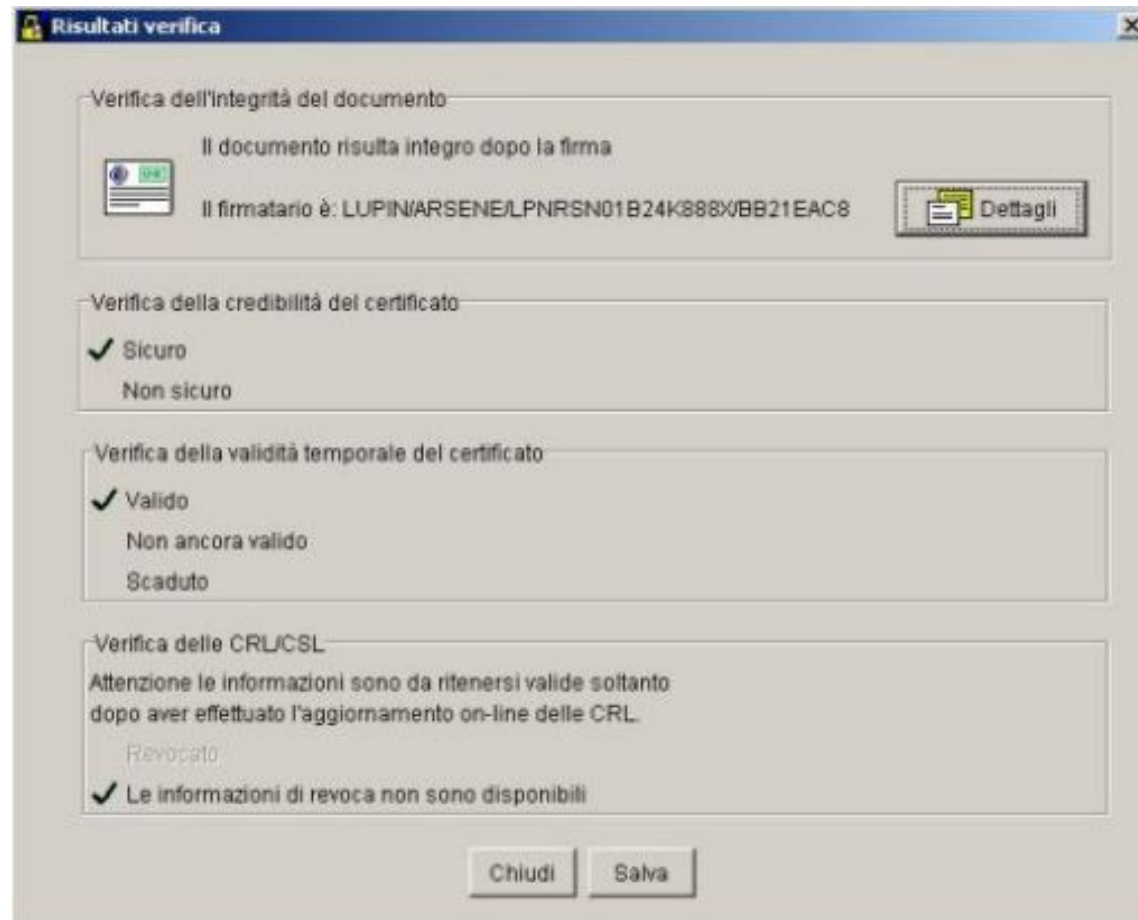
- Per verificare una firma, servono il certificato dell'autore e la catena di certificazione.
 - Teoricamente, tutti questi elementi sono disponibili **online**.
 - Per consentire la verifica offline, però, tutto viene incluso nel documento all'interno di una **busta PKCS#7**.
- Nel caso specifico, Firma&Cifra si fidava del certificato root incluso nella busta PKCS#7 — e addirittura lo importava nell'area di archiviazione sicura, trattandolo come affidabile.

L'exploit

1. Viene creato un falso certificato root con il nome di una CA reale. (es., PostECom)
2. Con quel certificato si emette un certificato utente falso (es. "Arsène Lupin")
3. Il documento viene firmato con il certificato utente fasullo.
4. I certificati falsi sono inclusi nella busta di firma (PKCS#7) allegata al documento.



L'exploit



TLS/SSL

Gli step di una connessione online sicura

1. **Client:** "Ciao!" (**Client Hello**)

Il client (ad esempio il browser) dice al server:

→ "Vorrei connettermi in modo sicuro. Ecco gli **algoritmi e i protocolli** che so usare (per esempio AES, ChaCha20, RSA, ecc.)."

2. **Server:** "Ciao anche a te!" (**Server Hello**)

Il server risponde:

→ "Perfetto, usiamo questo algoritmo che conosciamo entrambi. Ecco anche il mio **certificato digitale**, con la mia chiave pubblica, così puoi verificare che sono davvero io."

3. **Client:** "Controllo che tu sia chi dici di essere."

Il client **verifica la validità del certificato** (firma dell'autorità, scadenza, dominio).

→ In questa fase si usa crittografia asimmetrica (chiave pubblica e chiave privata).

Gli step di una connessione online sicura

4. **Client:** "Ho verificato che sei attendibile. Creiamo un segreto solo nostro."
Il client genera (o contribuisce a generare) una **chiave di sessione**, che sarà la chiave usata per la cifratura dei dati.
→ La invia in modo cifrato con la chiave pubblica del server, così solo il server può leggerla (ancora crittografia asimmetrica).
5. **Server:** "Ricevuto! Ora scegliamo come cifrare i dati."
Client e server concordano l'algoritmo di **cifratura simmetrica** (per esempio AES-256 o ChaCha20).
→ Tutta la comunicazione successiva userà quell'algoritmo insieme alla chiave di sessione appena scambiata.
6. **Entrambi:** "Da ora parliamo in segreto."
Da questo momento in poi, tutto ciò che viene inviato (testo, password, dati...) è protetto con **crittografia simmetrica**: stessa chiave su entrambi i lati, molto più veloce e adatta allo scambio continuo di informazioni.

Recap

- Funzioni di Hash: **integrità** e **anonimizzazione**.
- Crittografia Simmetrica: **confidenzialità con elevate performance**.
- Crittografia Asimmetrica: **confidenzialità con basse performance** e **autenticazione**.
- Certificati Digitali: **Autenticazione dell'identità** di un soggetto (es. un sito web) e associazione sicura tra una chiave pubblica e un'entità verificata.

Conclusioni

- La crittografia, se si usano algoritmi moderni, è affidabile e, anche se imperfetta, relativamente sicura!
 - Non si bypassa, si cercano altre strade.
 - La chiave è l'unica cosa che deve rimanere segreta. L'algoritmo è sempre da considerare noto.
- Tante nozioni:
 - Cifrari simmetrici/asimmetrici, funzioni di hash e certificati.
- Abbiamo visto due casi di studio di attacchi contro applicazioni crittografiche.
 - Questi attacchi avevano tutto a che fare con la sicurezza dei sistemi, senza nemmeno toccare gli algoritmi stessi.