

Sicurezza Informatica

Cifrari Asimmetrici



ElGamal
Goldwasser-Micali
Paillier



Diffie-Hellman Key Exchange

- Introduciamo lo scambio di chiavi DH in quanto il crittosistema di El Gamal si basa su di esso
- Alice e Bob alla fine del protocollo condivideranno una chiave segreta
 - La chiave potrebbe essere utilizzata per derivare una chiave di un cifrario a blocchi
- La sicurezza del protocollo si basa sulla difficoltà di calcolare il *logaritmo discreto* in gruppi finiti

DHKE

- Proposto nel 1976 da Whitfield Diffie e Martin Hellman
- Ampiamente utilizzato in Secure Shell (SSH), Transport Layer Security (TLS) e Internet Protocol Security (IPSec)
- DHKE è un protocollo di scambio chiavi e non è usato per cifrare
 - Si usa El Gamal

DHKE

Le operazioni sono in \mathbb{Z}_p^*
 α è un generatore del gruppo

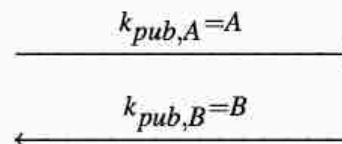
Diffie–Hellman Set-up

1. Choose a large prime p .
2. Choose an integer $\alpha \in \{2, 3, \dots, p-2\}$.
3. Publish p and α .

Diffie–Hellman Key Exchange

Alice
choose $a = k_{pr,A} \in \{2, \dots, p-2\}$
compute $A = k_{pub,A} \equiv \alpha^a \pmod{p}$

$$k_{AB} = k_{pr,A}^{k_{pub,B}} \equiv B^a \pmod{p}$$



Bob
choose $b = k_{pr,B} \in \{2, \dots, p-2\}$
compute $B = k_{pub,B} \equiv \alpha^b \pmod{p}$

$$k_{AB} = k_{pr,B}^{k_{pub,A}} \equiv A^b \pmod{p}$$

Ripetuto ogni volta che Alice e Bob vogliono scambiare una chiave

$$B^a \equiv (\alpha^b)^a \equiv \alpha^{ab} \pmod{p}$$

a chiave segreta di Alice

$$A^b \equiv (\alpha^a)^b \equiv \alpha^{ab} \pmod{p}$$

b chiave segreta di Bob

Esempio

Parametri

$p=29$

$\alpha=2$

Alice

choose $a = k_{pr,A} = 5$

$A = k_{pub,A} = 2^5 \equiv 3 \pmod{29}$

$A=3$



$B=7$



$k_{AB} = B^a \equiv 7^5 = 16 \pmod{29}$

Bob

choose $b = k_{pr,B} = 12$

$B = k_{pub,B} = 2^{12} \equiv 7 \pmod{29}$

$k_{AB} = A^b = 3^{12} \equiv 16 \pmod{29}$

$k_{pub,A}$ e $k_{pub,B}$ sono anche chiamate chiavi pubbliche effimere (*public ephemeral keys*) poiché non sono chiavi a lungo termine (*long term keys*). k_{AB} è una *chiave di sessione*

Osservazioni – 1

- Perché a e b sono scelti in $\{2, 3, \dots, p-2\}$?
 - Cosa succede se a (oppure b) è uguale ad 1?
 - Cosa succede se a (oppure b) è uguale a $p-1$?

Se $a = 1$, allora $k_{\text{pub},A} = \alpha$

Se $a = p-1$, allora $k_{\text{pub},A} = 1$

In entrambi i casi
la chiave privata di
Alice sarà rivelata

Teorema di Eulero

α deve essere un generatore del gruppo.

Oppure di un sottogruppo molto grande. Perché?

Per evitare che il calcolo del logaritmo discreto possa essere *semplice*

Osservazioni – 2

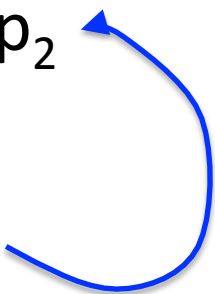
- Perché α è scelto in $\{2, 3, \dots, p-2\}$?
- Ovviamente α deve essere diverso da 1
- Dato che p è primo, si ha che $p-1 = 2 \cdot r$ per qualche intero r , quindi 2 è un divisore primo di $p-1 = \phi(p)$
- Calcoliamo $(p-1)^2 \bmod p$, si ha che $(p-1)^2 \equiv 1 \bmod p$
- Se r è primo ricordando che

$g \in \mathbb{Z}_n^*$ è un generatore di \mathbb{Z}_n^* se e solo se
$$g^{\phi(n)/d} \not\equiv 1 \pmod{n}$$

per ogni divisore primo d di $\phi(n)$

otteniamo che $\phi(p)/r = 2$, quindi $p-1$ non può essere un generatore

Generatore in \mathbb{Z}_p^* ?

- Lo possiamo generare facilmente se scegliamo il primo p in maniera opportuna
 - Scegliamo a caso due numeri primi p_1 e p_2
 - Settiamo $p = 2p_1p_2 + 1$
 - Se p è primo abbiamo finito, altrimenti
 - Scegliamo a caso un valore g in \mathbb{Z}_p^* e applichiamo
- 

Divisori di p :
 $2, p_1, p_2$

$g \in \mathbb{Z}_n^*$ è un generatore di \mathbb{Z}_n^* se e solo se
 $g^{\phi(n)/d} \not\equiv 1 \pmod{n}$
per ogni divisore primo d di $\phi(n)$

Il problema del logaritmo discreto

- È indicato con DLP (Discrete Logarithm Problem) ed è definito come segue

Definition 8.3.1 Discrete Logarithm Problem (DLP) in \mathbb{Z}_p^*
Given is the finite cyclic group \mathbb{Z}_p^ of order $p - 1$ and a primitive element $\alpha \in \mathbb{Z}_p^*$ and another element $\beta \in \mathbb{Z}_p^*$. The DLP is the problem of determining the integer $1 \leq x \leq p - 1$ such that:*

$$\alpha^x \equiv \beta \pmod{p}$$

Elemento primitivo = generatore

$$x = \log_{\alpha} \beta \pmod{p}$$

Ad esempio, calcolare x per cui $5^x \equiv 41 \pmod{47}$

Generalized Discrete Logarithm Problem

- Invece di usare \mathbb{Z}_p^* si può utilizzare qualsiasi gruppo ciclico finito G . La definizione di GDLP è la seguente

Definition 8.3.2 Generalized Discrete Logarithm Problem

Given is a finite cyclic group G with the group operation \circ and cardinality n . We consider a primitive element $\alpha \in G$ and another element $\beta \in G$. The discrete logarithm problem is finding the integer x , where $1 \leq x \leq n$, such that:

$$\beta = \underbrace{\alpha \circ \alpha \circ \dots \circ \alpha}_{x \text{ times}} = \alpha^x$$

Quali gruppi usare?

- Il gruppo \mathbb{Z}_p^* per un primo p oppure un suo sottogruppo
 - In genere si sceglie un safe prime $p=2\cdot q+1$ e si lavora sul sottogruppo moltiplicativo ciclico di ordine q
 - Ciò previene attacchi alla Pohlig–Hellman, se $p-1$ ha solo fattori primi piccoli, allora DLP è risolto in \mathbb{Z}_p^* efficientemente **dettagli in seguito**
- Sottogruppi moltiplicativi di campi di Galois $\text{GF}(2^m)$
- Gruppi di ordine $\geq 2^{160}$ generati da curve ellittiche

Problemi DL

Consideriamo un gruppo ciclico moltiplicativo finito G di ordine n con generatore g

- DLP/GDLP
 - Già esaminati
- Diffie-Hellman Problem (DHP)
 - Dati $a=g^x \in G$ e $b=g^y \in G$, con $a, b \in \mathbb{Z}_n$, trovare il valore $c \in G$ tale che $c=g^{xy}$
- Decisional Diffie-Hellman Problem (DDHP)
 - Dati $a=g^x$, $b=g^y$ e $c=g^z \in G$, con $a, b, c \in \mathbb{Z}_n$, stabilire se $x \cdot y \equiv z \pmod n$ oppure no

Sicurezza DHKE

- Un attaccante conosce
$$k_{\text{pub},A} = \alpha^a \bmod p \qquad k_{\text{pub},B} = \alpha^b \bmod p$$
- Vorrebbe conoscere $k_{A,B} = \alpha^{a \cdot b} \bmod p$
- Il problema di prima è DHP (Diffie-Hellman Problem)
- Tutti i problemi indicati nella slide precedenti sono considerati *difficili* da risolvere

Algoritmi per DLP

- **Algoritmi generici:** funzionano in qualsiasi gruppo ciclico, non sfruttano la struttura del gruppo
 - Brute-Force Search
 - Shanks' Baby-Step-Giant-Step Method
 - Pollard's Rho Method
 - Pohlig-Hellman Method
- La complessità di tempo
 - Per i primi tre dipende dall'ordine del gruppo
 - Per Pohlig-Hellman dipende dalla grandezza dei fattori primi dell'ordine del gruppo

} $O(|G|^{1/2})$

Algoritmi per DLP

- Per evitare l'attacco dei primi tre metodi si usano gruppi di ordine almeno 2^{160}
- Per evitare l'ultimo attacco il più piccolo fattore primo dell'ordine del gruppo deve essere grande almeno 2^{160}
- Algoritmi non generici
 - Funzionano solo in gruppi particolari (e.g., \mathbb{Z}_p^*)
 - Index Calculus Method

DLP in \mathbb{Z}_p^*

- Per utilizzare Diffie-Hellman in \mathbb{Z}_p^* si suggerisce di usare primi di lunghezza almeno pari a 2048 bit

Decimal digits	Bit length	Date
58	193	1991
68	216	1996
85	282	1998
100	332	1999
120	399	2001
135	448	2006
160	532	2007

Record più recenti

- Giugno 2014: 596 (lunghezza primo in bit)
- Giugno 2016: 768 (lunghezza primo in bit)
- Dettagli maggiori tramite la lista
 - NMBRTHRY Archives
 - NMBRTHRY@LISTSERV.NODAK.EDU
 - <https://listserv.nodak.edu/cgi-bin/wa.exe?AO=NMBRTHRY>

Record attuale: DLP in $GF(2^{9234})$ DLP in $GF(2^m)$

– R. Granger, T. Kleinjung, eJ. Zumbrägel, 31/01/2014

Subject: Discrete Logarithms in $GF(2^{9234})$

From: Jens Zumbrägel <[\[log in to unmask\]](#)>

Reply-To: Number Theory List <[\[log in to unmask\]](#)>, Jens Zumbrägel <[\[log in to unmask\]](#)>

Date: Fri, 31 Jan 2014 07:59:39 -0600

Content-Type: text/plain

Parts/Attachments:  [text/plain](#) (240 lines)

Dear Number Theorists,

We are pleased to announce a new record for the computation of discrete logarithms in finite fields. In particular, we were able to compute discrete logarithms in $GF(2^{9234})$ using about 400'000 core hours. To our knowledge the previous record was announced on 21 May 2013 in (a multiplicative subgroup of) the field $GF((2^{24})^{257})$ of 6168 bits [8].

The running time (in core hours) is as follows:

- relation generation 640 h (AMD: 6128 Opteron 2.0 GHz)
- linear algebra 258'048 h (Intel: Ivy Bridge 2.4 GHz)
- classical descent 134'889 h (Intel)
- Grobner basis descent 3'832 h (AMD)
- Pollard's rho 13 h (AMD)

totalling in 397'422 core hours.

Suggerimenti NIST


NIST SP 800-57, Part 1, Revision 4, Gennaio 2016

Security Strength	FFC (e.g., DSA, D-H)
≤ 80	$L = 1024$ $N = 160$
112	$L = 2048$ $N = 224$
128	$L = 3072$ $N = 256$
192	$L = 7680$ $N = 384$
256	$L = 15360$ $N = 512$

- FCC
 - Finite Field Cryptography
- L
 - Grandezza chiave pubblica
 - e.g., #bit del primo p
- N
 - Grandezza chiave privata
 - #bit esponente scelto da Alice e Bob. In genere si lavora su un sottogruppo (e.g., DSA)
dettagli in seguito

Attacco recente

- Nell'ottobre 2016 è stato pubblicato un attacco a DLP con p di 1024 bit

Subject: Discrete logarithms modulo a special 1024-bit prime.
From: Emmanuel Thomé <[\[log in to unmask\]](#)>
Reply-To: Number Theory List <[\[log in to unmask\]](#)>, Emmanuel Thomé <[\[log in to unmask\]](#)>
Date: Mon, 10 Oct 2016 16:40:08 +0200
Content-Type: text/plain
Parts/Attachments:  [text/plain](#) (34 lines)

We report the computation of discrete logarithms for a 1024-bit prime p using the special number field sieve (SNFS).

This calculation was made possible by the fact that we selected the prime p so that SNFS applies. However, this property is not visible from the prime p itself.

Attacco a DLP con 1024 bit

- Il primo p sembra casuale e $p-1$ ha un fattore primo di 160 bit, come raccomandato per DSA (Digital Standard Algorithm)
- Il primo p è stato costruito in modo che SNFS (special number field sieve) può fattorizzare p efficientemente
- Non è possibile accorgersi che p ha questa forma particolare
- In pratica è stata inserita una backdoor in DSA

J. Fried, P. Gaudry, N. Heninger, E. Thomé

[A kilobit hidden SNFS discrete logarithm computation](#)

Cryptology ePrint Archive: Report 2016/961

<http://eprint.iacr.org/2016/961>

Dettagli:

<http://caramba.inria.fr/hsnfs1024.html>

Snowden e NSA

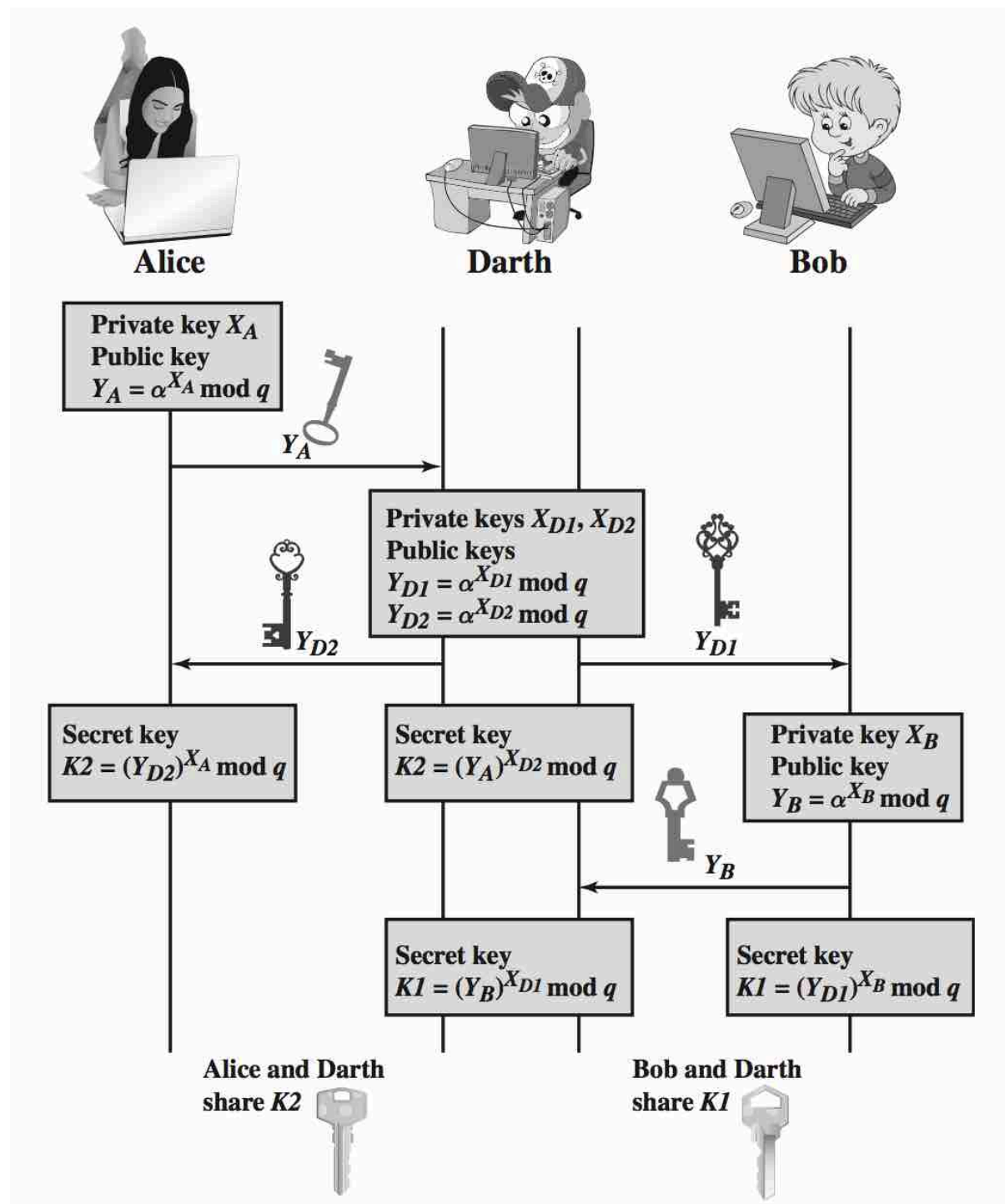
- In alcuni documenti NSA divulgati da Edward Snowden nel 2013 descrivono un progetto NSA “*SIGINT Enabling Project*” che tra gli obiettivi include

Influence policies, standards, and specification for commercial public key technologies

- Nei documenti si fa riferimento ad una backdoor inserita nel **PRNG Dual EC** standardizzato da NIST e ANSI

Già dal 2007 c'erano dei *sospetti* su PRNG Dual EC

Man in the middle attack



Man in the middle attack

- È possibile attaccare DHKE tramite un attacco Man in the Middle
- Il problema deriva dal fatto che le chiavi temporanee ($k_{\text{pub},A}$ e $k_{\text{pub},B}$) non sono autenticate
- Potremmo usare crittosistema a chiave pubblica per scambiare una chiave $k_{A,B}$ (detta chiave di sessione)
 - Sorge il problema della **Forward Secrecy**
 - Dobbiamo fidarci di chi genera la chiave

Forward Secrecy

- Un sistema possiede la proprietà di **Forward Secrecy**, se la compromissione di una chiave privata a lungo termine (in un certo punto del futuro) non compromette la comunicazione effettuata nel passato usando quella chiave
- Un crittosistema a chiave pubblica non gode della proprietà di **Forward Secrecy**
- DHKE (autenticato) gode della proprietà di FS

DHKE in pratica

- In molte applicazioni pratiche si usa un sottogruppo ciclico G di \mathbb{Z}_p^* . L'ordine di G sarà un primo p_1 con $p_1 \geq 2^{160}$
- Si generano due primi p_1 e p_2 con $p_1 \geq 2^{160}$ di modo che $p = 2 \cdot p_1 \cdot p_2 + 1$ sia primo e $\geq 2^{1024}$
- Dato un generatore α di \mathbb{Z}_p^* , un generatore g di G è calcolato come $g = \alpha^{\frac{p-1}{p_1}} = \alpha^{2 \cdot p_2}$
 - La chiave privata sarà un intero di almeno 160 bit

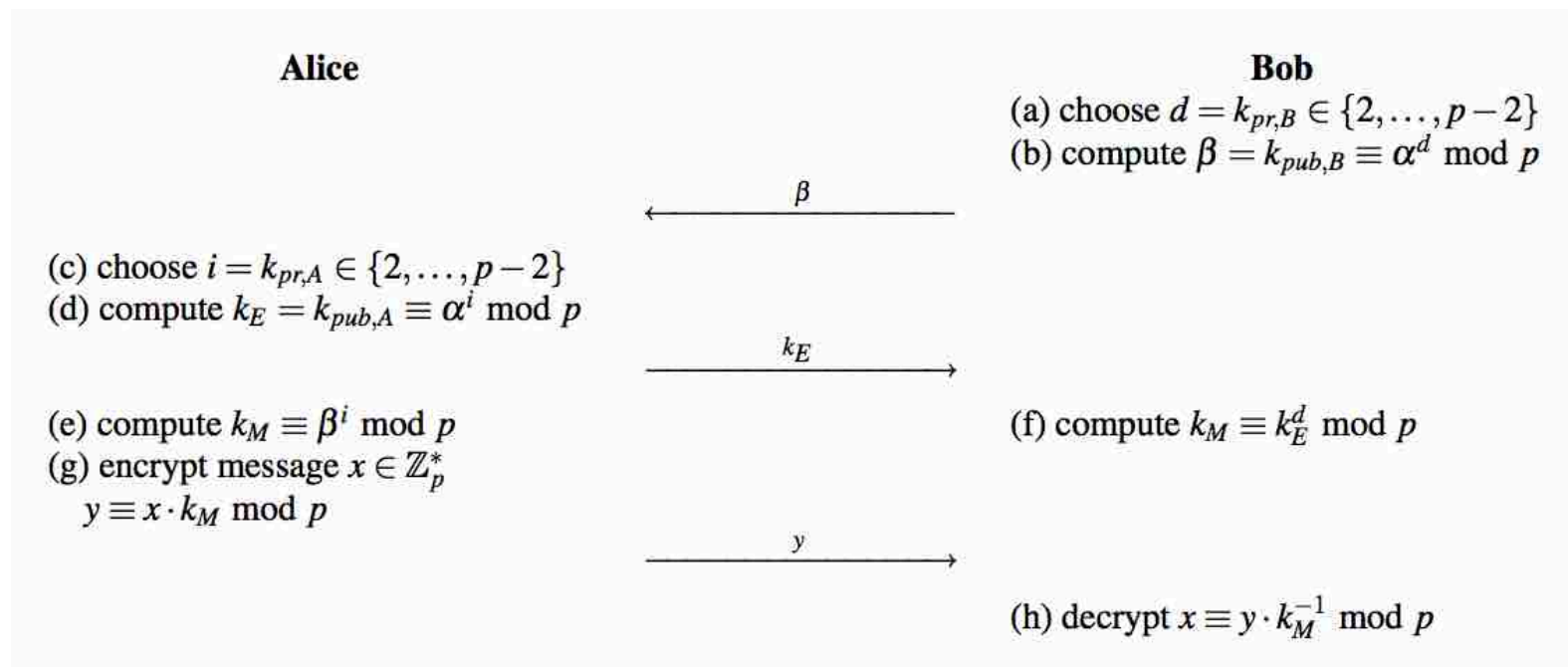
SCHEMA DI EL GAMAL

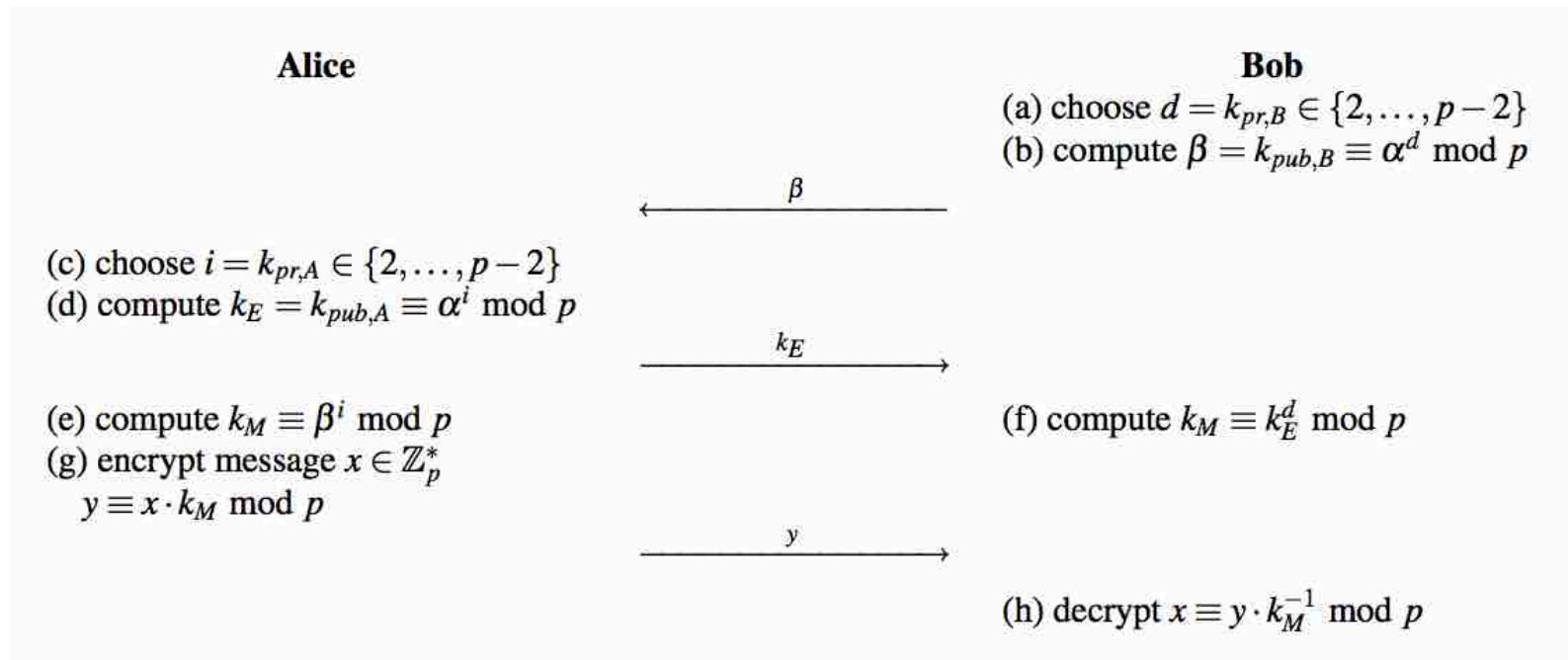


Introdotta da Taher Elgamal nel 1985

Principi alla base di El Gamal

- Una volta che una chiave di sessione è stata calcolata con DEKE, potremmo usare la chiave stessa per cifrare un messaggio m
 - Moltiplichiamo il messaggio m per $k_{A,B}$





- I passi (a)-(f) sono DEKE
 - La chiave pubblica di Bob non cambia
 - Alice deve cambiare le sue chiavi (i , k_E) ad ogni cifratura (E sta per ephemeral – esiste temporaneamente)
 - k_M è la masking key

k_M è scelta a caso in \mathbb{Z}_p^* quindi ogni testo cifrato è equiprobabile

Cifrario El Gamal

Alice

Operazioni in DHKE riorganizzate per risparmiare sulla comunicazione

choose $i \in \{2, \dots, p-2\}$
compute ephemeral key
 $k_E \equiv \alpha^i \mod p$
compute masking key
 $k_M \equiv \beta^i \mod p$
encrypt message $x \in \mathbb{Z}_p^*$
 $y \equiv x \cdot k_M \mod p$

$k_{pub} = (p, \alpha, \beta)$

←

(k_E, y)

→

Bob

choose large prime p
choose primitive element $\alpha \in \mathbb{Z}_p^*$
or in a subgroup of \mathbb{Z}_p^*
choose $k_{pr} = d \in \{2, \dots, p-2\}$
compute $\beta = \alpha^d \mod p$

compute masking key
 $k_M \equiv k_E^d \mod p$
decrypt $x \equiv y \cdot k_M^{-1} \mod p$

Perché funziona?

$$\begin{aligned}d_{k_{pr}}(k_E, y) &\equiv y \cdot (k_M)^{-1} \pmod{p} \\&\equiv [x \cdot k_M] \cdot (k_E^d)^{-1} \pmod{p} \\&\equiv [x \cdot (\alpha^d)^i] [(\alpha^i)^d]^{-1} \pmod{p} \\&\equiv x \cdot \alpha^{d \cdot i - d \cdot i} \equiv x \pmod{p}\end{aligned}$$

- Con El Gamal abbiamo un'espansione del testo cifrato
 - Un elemento del gruppo è cifrato con due elementi del gruppo
 - Non è diffuso quanto RSA perché è meno efficiente
- El Gamal è uno schema probabilistico

Sicurezza El Gamal

- Attacco passivo
 - L'attaccante ha accesso a p , α , $\beta = \alpha^d$, $k_E = \alpha^i$, $y = x \cdot \beta^i$
 - Il problema di calcolare x si riduce a DLP
- Attacchi passivi
 - La chiave pubblica non è autenticata
 - Attacco valido per qualsiasi schema a chiave pubblica che non usa i certificati per autenticare le chiavi
 - Se si riusa l'esponente i (stessa chiave effimera k_E) conoscendo un messaggio si calcola l'altro

Non cambiando l'esponente **i**

- La chiave k_E non cambia, quindi non cambia la chiave k_M
- Se (k_E, y_1) e (k_E, y_2) cifrano i messaggi x_1 e x_2 e l'attaccante riesce a calcolare x_1 , allora può ricavare k_M calcolando $k_M = y_1 \cdot x_1^{-1} \bmod p$
- Conoscendo k_M l'attaccante può ricavare x_2 calcolando $x_2 = y_2 \cdot k_M^{-1} \bmod p$

Malleabilità El Gamal

- Altro tipo di attacco attivo
- Un attaccante osservando (k_E, y) può sostituirlo con $(k_E, s \cdot y)$
- Il ricevente calcolerebbe

$$\begin{aligned} d_{k_{pr}}(k_E, sy) &\equiv sy \cdot k_M^{-1} \pmod{p} \\ &\equiv s(x \cdot k_M) \cdot k_M^{-1} \pmod{p} \\ &\equiv sx \pmod{p} \end{aligned}$$

Attacco simile a quello portato a Textbook RSA

Omomorfismo El Gamal

- Date due cifrature $c_1 = (k_{E1}, y_1)$ e $c_2 = (k_{E2}, y_2)$ di due messaggi x_1 ed x_2
- Moltiplicando c_1 e c_2 componente per componente otteniamo

$$(k_{E1} \cdot k_{E2}, y_1 \cdot y_2) = (\alpha^{i1} \cdot \alpha^{i2}, x_1 \cdot \beta^{i1} \cdot x_2 \cdot \beta^{i2}) = (\alpha^{i1+i2}, x_1 \cdot x_2 \cdot \beta^{i1+i2})$$

che corrisponde alla cifratura di $x_1 \cdot x_2$

(Gen, Enc, Dec) per El Gamal

$$(pk, sk) \leftarrow \text{Gen}(1^k)$$

$$pk = (p, \alpha, \beta = \alpha^d)$$

$$sk = d$$

p primo di k bit

$$d \leftarrow_R \mathbb{Z}_p^*$$

α generatore di \mathbb{Z}_p^*

$$c = (c_1, c_2) \leftarrow \text{Enc}(x, pk)$$

$$i \leftarrow_R \mathbb{Z}_p^*$$

$$c_1 = \alpha^i$$

$$c_2 = x \cdot \beta^i$$

$$x = \text{Dec}(c, sk)$$

$$\begin{aligned} x &= c_2 / c_1^d \\ &= c_2 \cdot (c_1^{-1})^d \end{aligned}$$

Aspetti computazionali El Gamal

- Generazione della chiave
 - Generazione di un primo p di 1024 bit (è meglio se si usa un primo di 2048 bit)
- Cifratura
 - Due elevamenti a potenza modulari ed una moltiplicazione modulare
- Decifratura
 - Un elevamento a potenza ($k_M = k_E^d$), inverso (k_M^{-1}) ed una moltiplicazione modulare
 - Applicando il Piccolo Teorema di Fermat si riduce a un elevamento a potenza +1 moltiplicazione modulare

Teorema di Eulero

$$k_E^{p-1} \equiv 1 \pmod{p}$$

$$\begin{aligned} k_M^{-1} &\equiv (k_E^d)^{-1} \pmod{p} \\ &\equiv (k_E^d)^{-1} k_E^{p-1} \pmod{p} \\ &\equiv k_E^{p-d-1} \pmod{p} \end{aligned}$$

Sicurezza El Gamal

- Si può provare che El Gamal
 - è IND-CPA sicuro
 - Prova basata su DDHP
 - è INC-CCA₁ sicuro
 - Si può provare sotto assunzioni *non standard*
 - **NON** è IND-CCA₂ sicuro
 - Facile da provare poiché El Gamal è omomorfo

El Gamal modificato

- Fujisaki and Okamoto nel 1999 hanno mostrato come modificare El Gamal per renderlo IND-CCA₂ sicuro
 - Hanno mostrato come trasformare uno schema IND-CPA in uno IND-CCA₂
- Lo schema ottenuto è leggermente meno efficiente dello schema di El Gamal

E. Fujisaki and T. Okamoto

[How to Enhance the Security of Public-Key Encryption at Minimum Cost](#)
Public Key Cryptography (PKC '99), LNCS 1560, Springer, pp 53-68, 1999.

H: funzione hash

|| *concatenazione*

Schema modificato

Si usa lo schema di El Gamal, per semplicità come input di Enc si fornisce anche il valore i . La funzione di cifratura è indicata con Enc_{EG}

$$\text{Enc}_{\text{EG}}(x, i, pk) = (\alpha^i, x \cdot \beta^i)$$

La nuova funzione di cifratura Enc è definita come

$$\begin{aligned}\text{Enc}(x, i, pk) &= \text{Enc}_{\text{EG}}(x || i, H(m || i), pk) \\ &= (\alpha^{H(m || i)}, x \cdot \beta^{H(m || i)})\end{aligned}$$

Si modifica anche l'uso della funzione di decifrazione. Si calcola

$$x' = \text{Dec}_{\text{EG}}(c)$$

Si verifica che

$$c = \text{Enc}_{\text{EG}}(x', H(x'), pk)$$

Se la verifica ha successo si ricava x a partire da

$$x' = x || i$$

Note

- Dobbiamo fare in modo che $x || i$ appartenga al gruppo su cui si basa El Gamal
- Nello schema modificato facciamo dipendere la randomness usata per la cifratura dal messaggio che cifriamo
- Lo schema che otteniamo è uno schema *plaintext aware*
- Lo schema ottenuto non è malleabile

SCHEMA DI GOLDWASSER-MICALI



Introdotta da Shafi Goldwasser e Silvio Micali nel 1982

Cifratura probabilistica

- Nozione introdotta da nel 1982 da Shafi Goldwasser e Silvio Micali
 - Erano studenti di PhD
- Vincitori del Turing Award 2012
 - *Per i contributi che hanno permesso di gettare le basi della teoria della complessità nel campo della crittografia e per aver sperimentato nuovi metodi per la verifica efficiente di dimostrazioni matematiche nel campo della teoria della complessità*
 - Il 20 maggio 2015 UNISA ha conferito a Silvio Micali la Laurea Magistrale ad Honorem in Informatica

Proprietà del crittosistema

Sicurezza Semantica

- Dato un testo cifrato, **qualsiasi informazione** che può essere calcolata, relativamente al testo in chiaro, in tempo polinomiale può essere calcolata efficientemente anche senza conoscere il testo cifrato stesso.
- La sicurezza basata sull'intrattabilità del problema residuosità quadratica
 - Decidere se un valore è un residuo quadratico modulo un numero composto ($N = p \cdot q$)

Sicurezza Semantica

- Concetto analogo al concetto di **sicurezza perfetta** (unconditional security) di Shannon
- Sicurezza perfetta significa che il testo cifrato non rivela alcuna informazione sul testo in chiaro; mentre, la sicurezza semantica implica che qualsiasi informazione rivelata non può essere **estratta** efficientemente

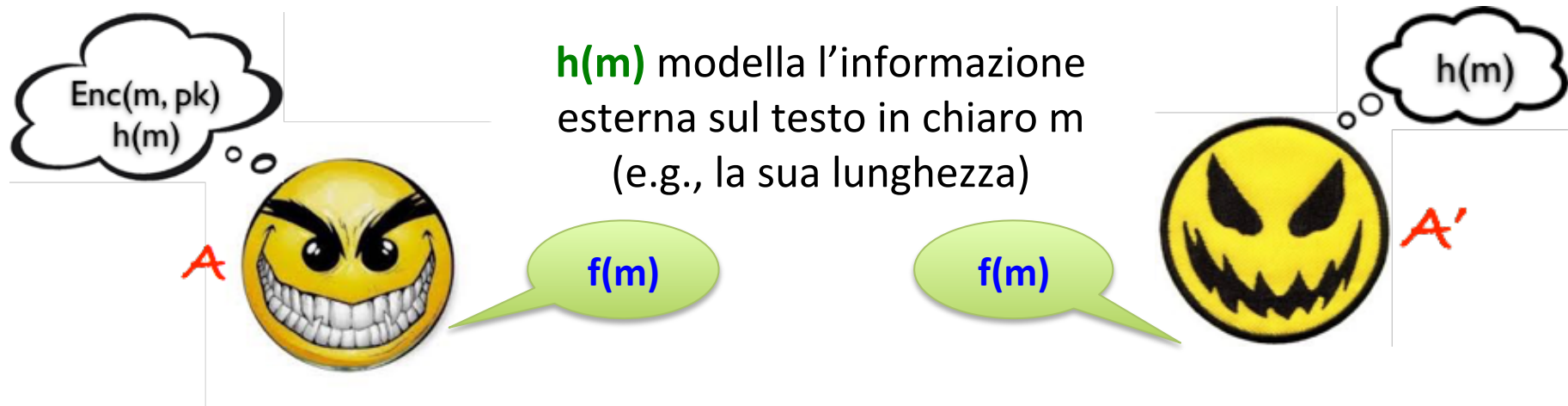
Possibile Definizione

- Un crittosistema è *semanticamente sicuro* se ogni algoritmo polinomiale probabilistico (PPTA) che riceve in input il messaggio cifrato C di un dato messaggio M e la lunghezza di M non può determinare alcuna informazione parziale sul messaggio con probabilità, non trascurabile (*non negligible*), maggiore di un qualsiasi PPTA che ha accesso solo alla lunghezza di M .

Definizione formale

L'avversario **A** conosce $\text{Enc}(m, pk)$ e $h(m)$ e tenta di dedurre il valore $f(m)$ (conoscenza acquisita sul messaggio m)

L'avversario **A'** tenta di dedurre il valore $f(m)$ conoscendo solo $h(m)$



Un crittosistema è *semanticamente sicuro* se per ogni avversario **A** esiste l'avversario **A'** per cui per ogni funzione f ed h esiste una funzione neglible negl tale che

$$|\text{Prob}(\mathbf{A}(1^k, \text{Enc}(m, pk), h(m)) = f(m)) - \text{Prob}(\mathbf{A}'(1^k, h(m)) = f(m))| < \text{negl}(k)$$

$\text{Enc}(k, pk)$ non serve per dedurre $f(m)$

Problemi con la definizione

- Non offre semplici strumenti per provare la sicurezza di crittosistemi
- Goldwasser e Micali introdussero il concetto di *indistinguibilità del testo cifrato* che provarono essere equivalente a sicurezza semantica

IND-CPA \equiv Semantic Security

Residuo quadratico

- Dati due numeri primi p e q , con N indichiamo il loro prodotto $N=p \cdot q$

Assunzione computazionale

- Dato (x, N) è difficile verificare se x sia un residuo quadratico modulo N

– Verificare se $x = y^2 \bmod N$

- Se si conosce la fattorizzazione di N , allora è facile verificare che x è un residuo quadratico modulo N

$$x \in \text{RQ}_n \iff \begin{cases} x^{(p-1)/2} \equiv 1 \pmod{p} \\ x^{(q-1)/2} \equiv 1 \pmod{q} \end{cases}$$

Schema di cifratura

- Codifichiamo 0 con un residuo quadratico
- Codifichiamo 1 con un residuo non quadratico
- Decodifica immediata (conoscendo la fattorizzazione di N)
- In realtà le cose non sono così semplici
 - Anche non conoscendo la fattorizzazione di N , verificare se x sia un residuo quadratico modulo N potrebbe essere facile
 - Esiste un algoritmo polinomiale per calcolare il **simbolo di Jacobi** che potrebbe indicare quando un residuo non è quadratico

Simbolo di Legendre

- Indicato con $\left(\frac{x}{p}\right)$ con p primo. Vale
 - 1, se x è un residuo quadratico modulo p
 - 0, se $x \equiv 0 \pmod{p}$
 - -1, se x non è un residuo quadratico modulo p
- In base alla definizione del simbolo di Legendre, si ha che

$$\left(\frac{x}{p}\right) \equiv x^{(p-1)/2} \pmod{p}$$

Simbolo di Jacobi

- Generalizzazione del simbolo di Legendre
- Dato $N = p_1^{a_1} p_2^{a_2} \cdots p_k^{a_k}$, per qualsiasi intero x , il simbolo di Jacobi è il prodotto dei simboli di Legendre corrispondenti ai fattori primi di N

$$\left(\frac{x}{N}\right) = \left(\frac{x}{p_1}\right)^{a_1} \left(\frac{x}{p_2}\right)^{a_2} \cdots \left(\frac{x}{p_k}\right)^{a_k}$$

$$\text{per } N=p \cdot q \quad \left(\frac{x}{N}\right) = \left(\frac{x}{p}\right) \cdot \left(\frac{x}{q}\right)$$

Proprietà di $\left(\frac{x}{N}\right)$

$$\left(\frac{0}{N}\right) = 0$$

$$\left(\frac{1}{N}\right) = 1$$

$$\left(\frac{2}{N}\right) = \begin{cases} -1 & \text{se } N \equiv 3 \text{ oppure } 5 \pmod{8} \\ 1 & \text{se } N \equiv 1 \text{ oppure } 7 \pmod{8} \end{cases}$$

$$\left(\frac{x \cdot y}{N}\right) = \left(\frac{x}{N}\right) \left(\frac{y}{N}\right)$$

$$\left(\frac{x}{N}\right) = \left(\frac{y}{N}\right) \text{ se } x \equiv y \pmod{N}$$

$$\left(\frac{x}{N}\right) = (-1)^{\frac{x-1}{2} \frac{N-1}{2}} \left(\frac{N}{x}\right)$$

Algoritmo polinomiale per il calcolo di $\left(\frac{x}{N}\right)$

```
def jacobi(x,n):  
    if x == 0:  
        return 0  
    if x == 1:  
        return 1  
  
    if x == 2:  
        n8 = n%8  
        if n8 == 3 or n8 == 5:  
            return -1  
        else:  
            return 1  
  
    if x%2 == 0:  
        return jacobi(2,n) * jacobi(x/2,n)  
  
    if x >= n:  
        return jacobi(x%n,n)  
  
    if x%4 == 3 and n%4 == 3:  
        return -jacobi(n,x)  
    else:  
        return jacobi(n,x)
```

Problema difficile?

- Dato (x, N) , con $N=p \cdot q$, è difficile verificare se x sia un residuo quadratico modulo N ?
- Dato l'algoritmo di prima, dipende da come è *fatto* x
- Se $\left(\frac{x}{N}\right) = -1$ allora siamo sicuri che x **NON** è un residuo quadratico modulo N
 - Risulta che
 - x non è un residuo quadratico modulo p oppure
 - x non è un residuo quadratico modulo q

Osservazione

$$x^{\frac{p-1}{2}} \equiv 1 \pmod{p} \quad x^{\frac{q-1}{2}} \equiv 1 \pmod{q} \quad \left(\frac{x}{N}\right) = 1 \quad \checkmark$$

$$x^{\frac{p-1}{2}} \equiv 1 \pmod{p} \quad x^{\frac{q-1}{2}} \equiv -1 \pmod{q} \quad \left(\frac{x}{N}\right) = -1 \quad \times$$

$$x^{\frac{p-1}{2}} \equiv -1 \pmod{p} \quad x^{\frac{q-1}{2}} \equiv 1 \pmod{q} \quad \left(\frac{x}{N}\right) = -1 \quad \times$$

$$x^{\frac{p-1}{2}} \equiv -1 \pmod{p} \quad x^{\frac{q-1}{2}} \equiv -1 \pmod{q} \quad \left(\frac{x}{N}\right) = 1 \quad \checkmark$$

Un valore x **non residuo quadratico** modulo N che ha simbolo di Jacobi uguale ad 1 è chiamato **pseudo-quadratico**

Come calcoliamo gli elementi di RQ_N ?

- Dato $N=p \cdot q$, per calcolare un residuo quadratico x scegliamo un valore a caso $r \in \mathbb{Z}_N^*$ e poniamo $x = r^2 \bmod N$
- Dati p e q , per calcolare un residuo non quadratico x con simbolo di Jacobi pari ad 1, scegliamo x a caso in \mathbb{Z}_N^* e verifichiamo che

$$x^{(p-1)/2} \equiv -1 \pmod{p} \text{ e } x^{(q-1)/2} \equiv -1 \pmod{q}$$

Se $p \equiv 3 \pmod{4}$ e $q \equiv 3 \pmod{4}$, allora $N-1$ è un residuo non quadratico con simbolo di Jacobi pari ad 1

Lo schema

- Si cifrano stringhe binarie bit a bit
- Cifratura
 - Cifriamo 1 con un residuo non quadratico con simbolo di Jacobi uguale ad 1
 - Residuo *pseudo-quadratico*
 - Cifriamo 0 con un residuo quadratico
- Decifrazione immediata

GM Encryption

- $N=p \cdot q$ con p e q numeri primi di k bit
- Chiave pubblica: (x, N) dove x è un residuo non quadratico modulo N con Jacobi pari ad 1
- Chiave segreta: (p, q, N)
- Cifratura di una stringa $B = b_1 b_2 \dots b_m$

Per $i=1, \dots, m$

Scegli un $r \in \mathbb{Z}_N^*$

Se $b_i = 1$

Poni $e_i = x \cdot r^2 \bmod N$

altrimenti

Poni $e_i = r^2 \bmod N$

Il prodotto di un residuo quadratico per un residuo non quadratico è un residuo non quadratico

Proprietà GM

- Poco efficiente
 - Un bit è codificato con $\log |N|$ bit
- Possiede proprietà omomorfe
 - Se c_1 e c_2 sono le cifrature dei messaggi m_1 ed m_2 , allora $c_1 \cdot c_2 \bmod N$ è una cifratura di $m_1 \otimes m_2$
 - Il simbolo \otimes indica l'operatore XOR

SCHEMA DI PAILLIER



Introdotta da Pascal Paillier nel 1999

Schema di Paillier

- Schema probabilistico con proprietà omomorfe
- Date le cifrature $\text{Enc}(\text{pk}, m_1)$ ed $\text{Enc}(\text{pk}, m_2)$ di due messaggi m_1 ed m_2 ed una costante c , senza conoscere sk , possiamo calcolare efficientemente
 - $\text{Enc}(\text{pk}, m_1 + m_2)$
 - Calcolando $\text{Enc}(\text{pk}, m_1) \cdot \text{Enc}(\text{pk}, m_2)$
 - $\text{Enc}(\text{pk}, c \cdot m_1)$
 - Calcolando $\text{Enc}(\text{pk}, m_1)^c$

Generazione chiave

- Si generano due numeri primi p e q
- Si pone $n = p \cdot q$
- $\varphi(n) = (p-1)(q-1)$ e $\lambda(n) = \text{mcm}(p-1, q-1)$
 - Per alleggerire la notazione si usa $\lambda(n)$ al posto di λ
- Si considerano i gruppi \mathbb{Z}_n e $\mathbb{Z}_{n^2}^*$
- Si sceglie $g \in \mathbb{Z}_{n^2}^*$ di ordine $\alpha \cdot n$
- Chiave pubblica (g, n)
- Chiave privata (p, q) o equivalentemente λ

Cifrare/Decifrare

- **Enc**

- Si sceglie un valore a caso r in \mathbb{Z}_n
- $c = \text{Enc}(m, pk) = g^m \cdot r^n \bmod n^2$

- **Dec**

- $m = L(c^\lambda \bmod n^2) \cdot L(g^\lambda \bmod n^2)^{-1} \bmod n$

- La funzione $L(x)$ è definita come quoziente di $x-1$ diviso n

- Il più grande intero $v \geq 0$ che soddisfa la relazione
 $(x-1) \geq v \cdot n$

Dettagli

- Generazione di g
 - Si sceglie a caso un $g \in \mathbb{Z}_{n^2}^*$ e si verifica che $\text{mcd}(L(g^\lambda \bmod n^2), n) = 1$
- Il valore $\mu = L(g^\lambda \bmod n^2)^{-1} \bmod n$ può essere precalcolato, la chiave privata potrebbe essere (λ, μ)
- Se p e q sono di lunghezza equivalente, allora
$$g = n+1, \quad \lambda = \varphi(n) = (p-1)(q-1), \quad \mu = \lambda^{-1}$$

Sicurezza dello schema

- Lo schema è IND-CPA ma non è IND-CCA₂
 - Lo schema è omomorfo, quindi malleabile
 - Si può provare che lo schema è IND-CCA₁
- Apportando una modifica allo schema lo si può rendere IND-CCA₂ sicuro
 - Si combina l'hashing del messaggio con la randomness utilizzata

Riferimenti

Christof Paar and Jan Pelzl

Understanding Cryptography

Capitolo 8 **Public-Key Cryptosystems Based on the Discrete Logarithm Problem**

(tranne Paragrafo 8.2 **Some Algebra**)

Alfred Menezes, Paul van Oorschot, and Scott Vanstone

Handbook of Applied Cryptography

CRC Press, 1996

Paragrafo 8.7 **Probabilistic public-key encryption**