CRYPTO - A PAROLE

2021

Di seguito, le lezioni del 2021 dalla Lecture 1 alla Lecture 17, con integrazioni dagli appunti del 2022.

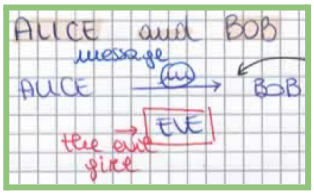
Lecture 1

[2]: 2 [1]: 2 [7]: 1.1

- Introduction to the course.
- Secure communication: message confidentiality and authenticity.
- Symmetric encryption and perfect secrecy.
- Equivalent notions of perfect secrecy.
- The one-time pad and Shannon's impossibility result.

Introduction and secure communication

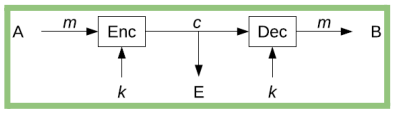
Tra le più importanti componenti di un secure (digital) system, la crittografia è usata per garantire comunicazioni sicure (a discapito di un ADVERSARY considerato cattivo). Veniva usata principalmente nel campo militare, ma negli ultimi anni il suo approccio è cambiato da essere più creativo a più ingegneristico. Protocolli basati su crittografia quali TLS vengono usati per costruire comunicazioni sicure in Internet. Nell'approccio ingegneristico, perciò, la creazione di qualcosa di nuovo non è abbastanza: da un'astrazione matematica bisogna modellare la chiave e creare una formal proof in modo tale che la chiave soddisfi le proprietà (bullet proof).



I due "protagonisti" del corso sono Alice e Bob, che scambiano un messaggio (m). Il nostro ADVERSARY viene interpretato da Eve, che vuole intercettare il messaggio. In questo caso, la prima regola è essere pessimisti: se qualcosa di cattivo può succedere, succederà (worst case). Quindi, considereremo sempre il peggior caso possibile, ovvero che Eve riesca a leggere il messaggio (CONFIDENTIALITY) e modificarlo (INTEGRITY). Da qui si evince che:
MESSAGE CONFIDENTIALITY + MESSAGE INTEGRITY = SECURE COMMUNICATION .
Esistono due modi per avere una SECURE COMMUNICATION: il primo approccio è mediante symmetric cryptography, il secondo con asymmetric cryptography. Questi due permetteranno di risolvere i due sottoproblemi della SECURE COMMUNICATION: confidentiality ed integrity.

Symmetric Cryptography

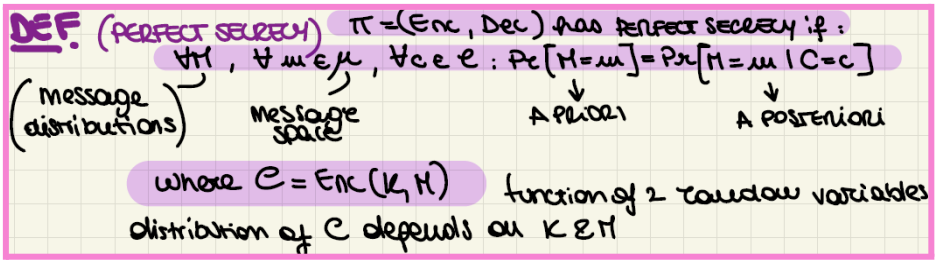
In symmetric cryptography, si ha la stessa chiave sia per l'encryption sia per la decryption. Il nostro primo random problem ha come PLAN di nascondere il messaggio. Questo avviene creando una primitiva con Enc e Dec. Il messaggio passerà per l'encryption. Il risultato dell'encryption è il ciphertext, eseguito su un plaintext con un algoritmo chiamato cipher. Questo ciphertext passerà poi per Dec per essere decifrato. Se Enc e Dec sono giusti, il risultato tornerà ad essere m.
Abbiamo raggiunto il nostro PLAN? Se consideriamo Enc e Dec privati e conosciuti solo ad Alice e Bob si: questo però richiede troppo effort. Inoltre potrebbero essere scoperti e pubblicati o persi.
NEW PLAN: nascondere il messaggio senza che la secrecy del sistema dipenda dalla sicurezza del codice (ovvero Enc e Dec). Questo avviene tramite l'aggiunta di una chiave [Def 1 - π is correct if $\forall k, \forall m \rightarrow Dec(k, Enc(k, m)) = m$] condivisa tra Alice e Bob e generata randomicamente ($2^{-\lambda}$ probabilità che venga scelta).

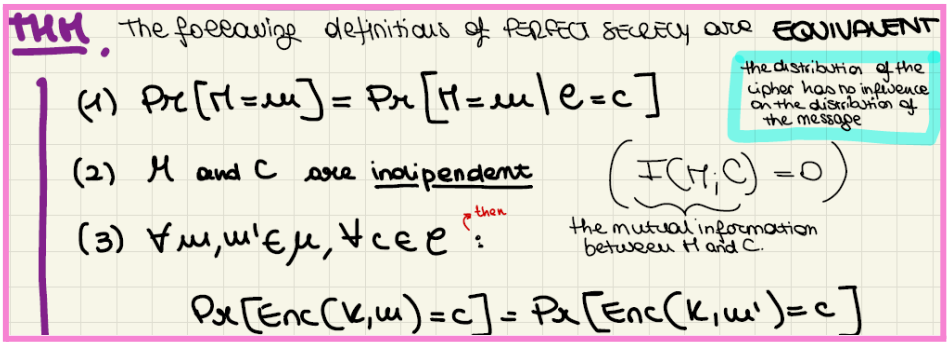


- Def 1 - π is correct if $\forall k, \forall m \rightarrow Dec(k, Enc(k, m)) = m$

Perfect secrecy

Come definiamo la secrecy? Noi vogliamo che il ciphertext non riveli niente del plaintext e che la definizione valga per ogni distribuzione M di messaggi [Def 2 - Perfect Secrecy] [Thm 1 - Equivalent notions of perfect secrecy].

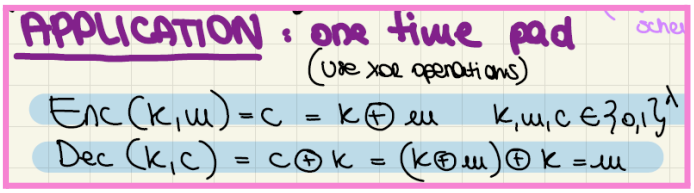


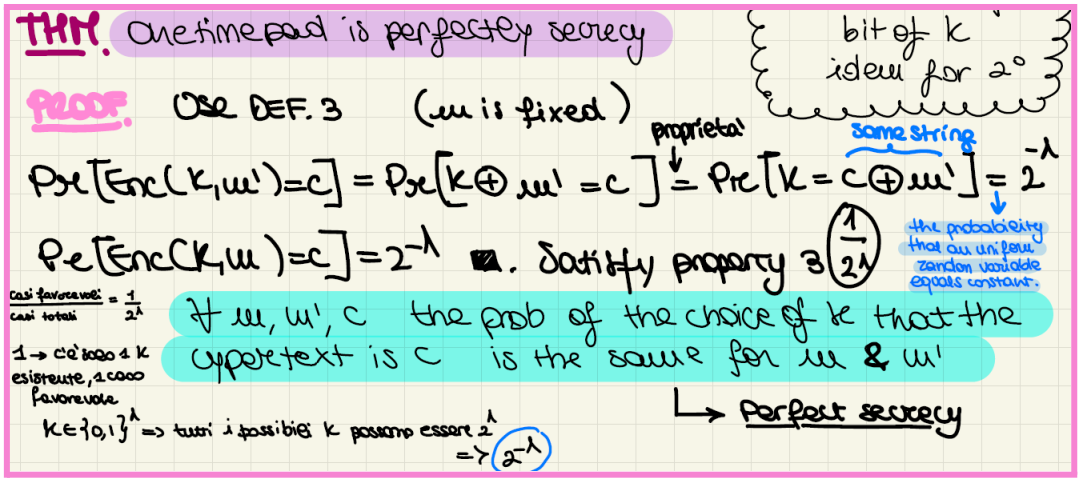


- Def 2 - Perfect Secrecy
- Thm 1 - Equivalent notions of perfect secrecy (+ proof)

One time pad (OTP)

Un'applicazione di perfect secrecy è One time pad [Thm 2 - One Time Pad is perfectly secret].





- Thm 2 - One Time Pad is perfectly secret (+ proof)

Abbiamo finalmente risolto il nostro PLAN, ma sorgono delle limitazioni:

- $|k| = |m|$, quindi bisogna conoscere in anticipo la lunghezza del messaggio e, se questo è lungo, allora anche la chiave sarà lunga;
- One Time indica che esiste solo un ciphertext per la stessa chiave (it's impossible to use the same key twice).

Allora ci si chiede, prima di formulare un nuovo PLAN, esiste una encryption che risolve questi problemi? La risposta è no (limitations hold for all perfectly secret schemes). Questo viene provato con il teorema di Shannon (Lecture 2) (it's not in the theorem, but it can be proved that, for any perfectly secret encryption, one key corresponds to one cipher: the scheme cannot be used twice, because no scheme can achieved the two-time secure encryption, so one key for ciphertext).

Exercise. Define two-time secure encryption. Prove OTP is not two-time secure.

PLAN [per tutto il symmetric encryption]: trovare uno schema che permetta a due persone di scambiarsi una chiave corta che sia valida per l'intera vita, ovvero superare Shannon (ovvero trovare uno schema in cui la chiave non "dipenda" dalla lunghezza del messaggio → avere una chiave corta).

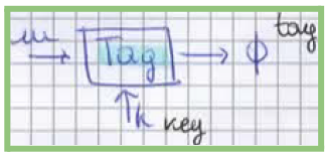
Lecture 2

[2]: 4.6 [7]: 1.2, 1.3

- ➢ Message Authentication Codes (MACs).
- ➢ Definition of statistically-secure (one-time) MACs.
- ➢ Pair-wise independent hashing: Definition and construction using modular arithmetic.
- ➢ Application to statistically-secure message authentication.
- ➢ Limits of statistically-secure MACs.

Shannon's theorem

Il teorema di Shannon [Thm 3 - Shannon] afferma che dato uno schema perfectly secret $\Pi = (Enc, Dec)$, allora $|\mathcal{K}| \geq |\mathcal{M}|$ (the length of the key must be at least the length of the message).

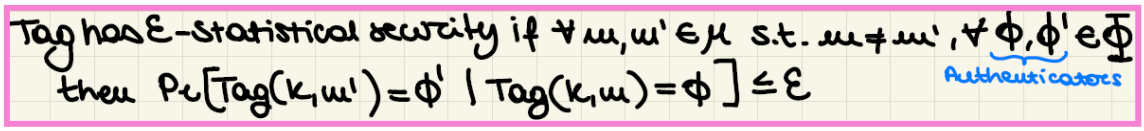


- Thm 3 - Shannon (+ proof)

Message Authentication Codes (MACs)

Statistically-secure (one-time) MACs

Per risolvere l'integrity problem (nella crittografia simmetrica), si crea un Message authentication code (MAC), utilizzando una primitiva chiamata Tag che prende in input una chiave ed un messaggio. Come ogni volta, abbiamo definito un modello con questa primitiva, ora il nostro PLAN è di provare che regge le proprietà (integrity). Definiamo quindi l'information-theoretic security (o statistically secure, poiché definita con termini probabilistici) per un certo Tag [Def 3 - Information Theoretic MAC].



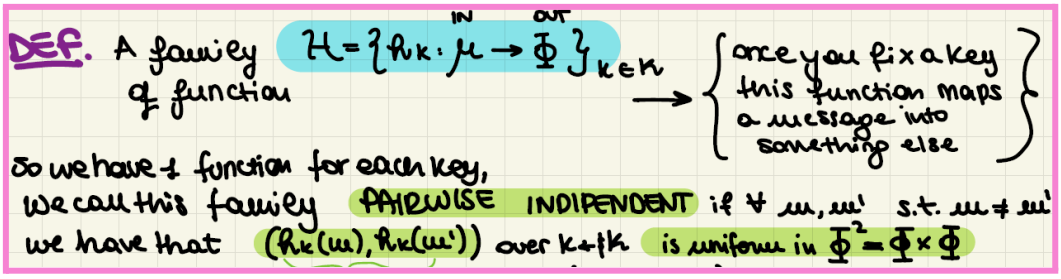
- Def 3 - Information Theoretic MAC

[telegram]: 2.1
A security problem arises when someone, having a signed message (m1; t1) where the key k used for tagging is unknown, is able to efficiently sign a different message (m2; t2) such that verication under the same key yields a success. This action is called a forgery; if Eve is effectively able to forge valid signatures, she can indeed impersonate either Alice or Bob at will. The desired property of an authentication scheme thus becomes to be resistant, if not immune, to such attacks; in one word, the scheme is unforgeable.
[7]: 1.2
Information theoretic security is also called unconditional security. Later we'll see conditional security, based on computational assumptions.

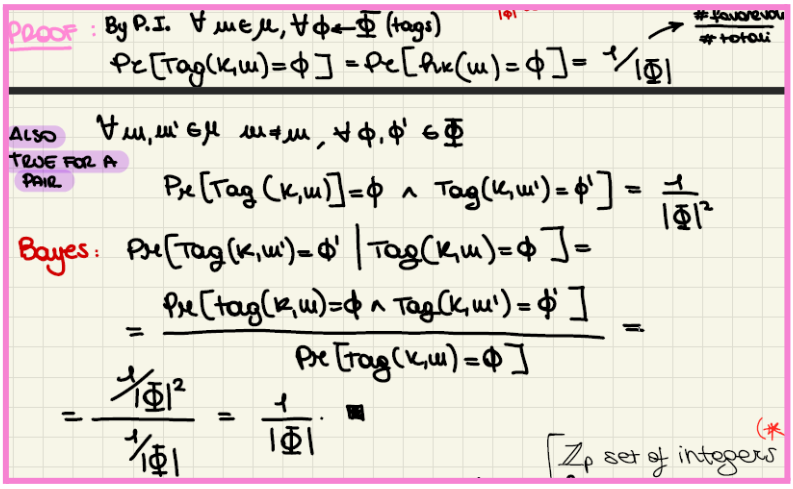
Modellato il nostro schema, il nuovo PLAN è migliorarlo. Come? Ci chiediamo se ε possa essere 0: ciò è intrinsecamente impossibile, perché un avversario potrà sempre provare ad ottenerlo. Allora, il nuovo PLAN diventa avere un buon Tag con piccola ε. Si parte quindi dalla (Def 3), che però è valida solo se si ha un Tag per un messaggio. La definizione non è abbastanza forte per reggere la stessa chiave per creare il Tag di due messaggi.

Pairwise Independent Hashing

Si costruisce quindi una famiglia di funzioni, tramite il PAIRWISE INDEPENDENT HASHING (tool), dalla definizione [Def 4 - Pairwise Independent Hashing].



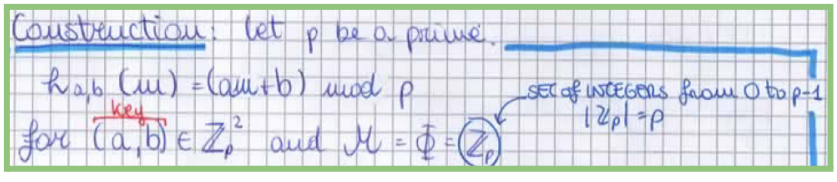
Tramite [Thm 4 - Any pairwise independent function is $\frac{1}{|\Phi|}$ - Statistically Secure] proviamo che ogni funzione pairwise independent è $\frac{1}{|\Phi|}$ - Statistically Secure. Dimostriamo il teorema utilizzando (Def 4) e il teorema di Bayes.

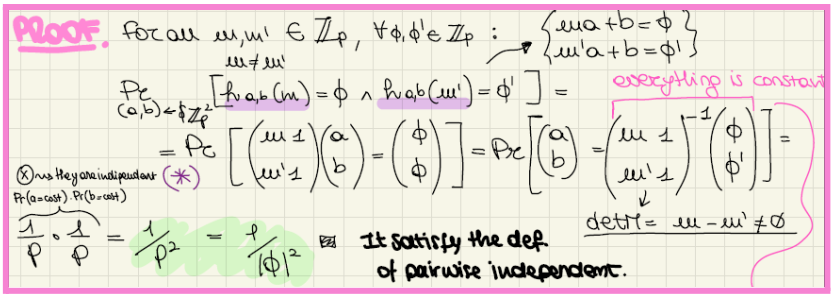


Adesso, la definizione di pairwise independence è più forte di quella di Tag perchè regge anche su due messaggi distinti (ma non per più di due). I due messaggi devono essere distinti, altrimenti l'hash sarebbe lo stesso e la distribuzione non sarebbe uniforme.

- Def 4 - Pairwise Independent Hashing
- Thm 4 - Any pairwise independent function is $\frac{1}{|\Phi|}$ - Statistically Secure (+ proof)

Costruiamo quindi un esempio di famiglia pairwise independent [Construction] usando l'aritmetica modulare e dimostriamo che questa è pairwise independent [Lemma - The functions in the family $\mathcal{H}$ (Construction) are pairwise independent].





- (Construction +) Lemma - The functions in the family $\mathcal{H}$ (Construction) are pairwise independent (+ proof)

Limits of statistically-secure MACs.

Nella nostra costruzione rimangono però le limitazioni sulla chiave, ovvero che può essere usata solo una volta (One Time) e che la sua lunghezza deve essere il doppio di quella del messaggio. Si può avere una costruzione migliore? Definiamo un teorema per cui la grandezza della chiave cresce con la grandezza del messaggio [Thm 5 - Any t-time $2^{-\lambda}$ - statistically secure MAC has key of size $(t+1)\lambda$].

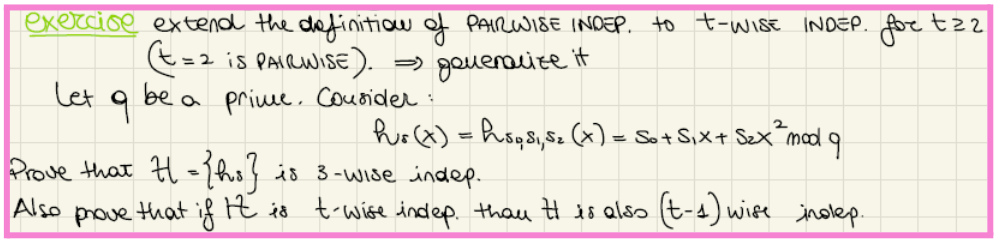
- Thm 5 - Any t-time $2^{-\lambda}$ - statistically secure MAC has key of size $(t+1)\lambda$

Lecture 3

[4]: 6 [1]: 3.5

- ➢ The problem of randomness extraction, and definition of min-entropy.
- ➢ The von Neumann extractor.
- ➢ Impossibility of seedless extractors for min-entropy sources.
- ➢ Definition of seeded extractors.
- ➢ Leftover-hash lemma: Statement and proof.

Exercise.



Randomness Extraction

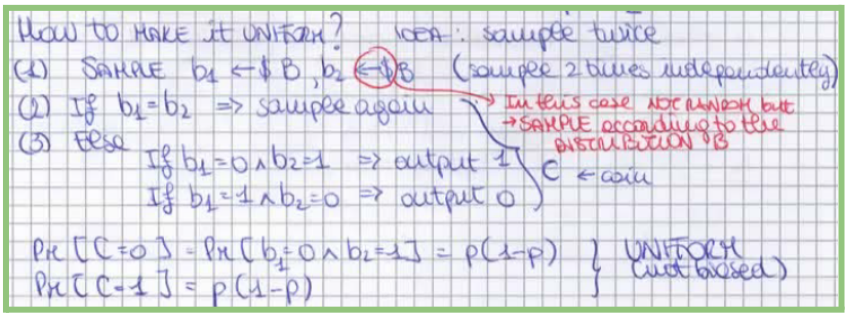
La randomicità è cruciale in crittografia:

- per generare chiavi uniformi;
- per gli algoritmi di crypto (in genere sono randomizzati e non deterministici).

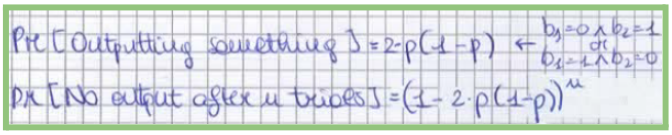
In natura però e nei processi fisici, non esiste la randomicità pura, ma solo qualcosa che ci si avvicina. Come facciamo a trasformare qualcosa che si avvicina al random in un random puro (how to turn a “somewhat random” source in a “truly random” source)?

Von Neumann extractor

Un esempio è il Von Neumann Extractor: sia B una biased distribution (quindi, appunto, non uniforme, $Pr[B=0]=p$, $Pr[B=1]=1-p$, $p<1/2$), è possibile renderla uniforme (vedi figura sotto).



Distribution is ok (uniform), but we are always sampling again: we have to ensure that at some point we will produce an output.

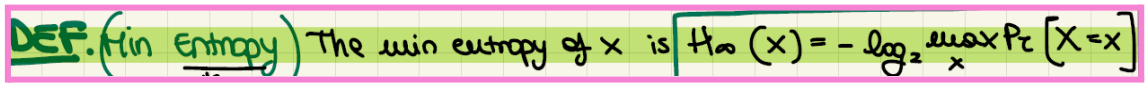


In real life (in real physical processes), we don't have these assumptions on the source B. In practice, physical processes produce X which is "unpredictable". We have to formalize what it means for a random variable to be unpredictable.

[1]: 3.5 (Estrattori di randomicità)
Un approccio è quello di affidarsi ad alcuni fenomeni fisici la cui natura sembra impredicibile. Purtroppo neanche tali sorgenti sono veramente casuali, ma tipicamente presentano un qualche grado di correlazione (per questo motivo si parla di sorgenti deboli di randomicità). Ha senso allora chiedersi se esiste una procedura per estrarre randomicità: data una sorgente debole di randomicità vogliamo estrarre da essa quanta più randomicità vera possibile. La risposta a questa domanda ha portato al concetto di estrattore.

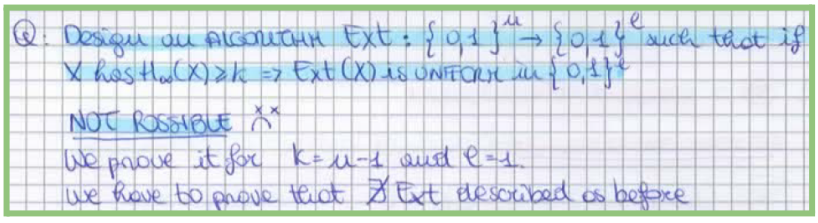
Definition of min-entropy and seedless extractors

PLAN: formalizzare cosa significa per una variabile random X essere imprevedibile e in che modo posso avere una sorgente non dipendente dall'estrattore. Definiamo quindi la MIN-ENTROPY [Def 5 - Min Entropy]: questa misura l'impredicibilità di X ("Worst chance to guess X in the worst case"). Maggiore sarà questa, maggiore sarà l'impredicibilità.



- Def 5 - Min Entropy

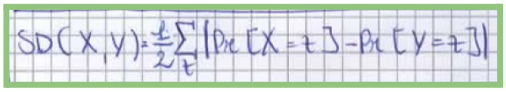
Applicandolo al caso reale, possiamo dimostrare che non esiste un estrattore in grado di ritornare una distribuzione uniforme a partire da sorgenti deboli di randomicità.



Definition of seeded extractors

Consideriamo adesso una famiglia di estrattori, di tipo seeded extractor [Def 6 - Seeded Extractor]. Dobbiamo sempre ricordarci che, sebbene non si riesca a raggiungere la randomicità pura, anche vicino alla randomicità uniforme è accettabile. Consideriamo

dunque un seed come un piccolo seme di randomicità, output quindi di un processo fisico. Questo deve essere più corto della randomicità che si vuole produrre affinché l'estrattore funzioni. Il seed non è necessario che sia segreto. La definizione di seeded extractor utilizza la definizione di statistical distance, per affermare che non esiste adversary in grado di distinguere una distribuzione puramente uniforme da una output del nostro estrattore con probabilità maggiore di ε (solo una piccola chance).



- Def 6 - Seeded Extractor

[1]: Definizione A.3 (Distanza statistica)

Definizione A.3 (Distanza statistica). Date due variabili aleatorie X_0, X_1 con valori in $\mathcal{X}$, la loro distanza statistica è:

$$\Delta(X_0, X_1) = \frac{1}{2} \sum_{x \in \mathcal{X}} |\mathbb{P}[X_0 = x] - \mathbb{P}[X_1 = x]|.$$

Leftover-hash lemma: statement and proof

Enunciamo quindi il Leftover Hash Lemma [Thm 6 - Leftover Hash Lemma], che afferma che ogni pairwise independent hash function (Def 4) è un buon estrattore. Per provare il teorema abbiamo bisogno di definire la COLLISION PROBABILITY tramite un lemma [Lemma - Collision Probability], in quanto il primo passo della dimostrazione prevede che sia computata la statistical distance come funzione della COLLISION PROBABILITY (usando la definizione di min-entropy (Def 5) e di pairwise independent (Def 4) per lavorare con le probabilità). Poi, viene usato l'upper bound della COLLISION PROBABILITY per avere un upper bound della statistical distance.

- Thm 6 - Leftover Hash Lemma (+ proof) + Lemma - Collision Probability (+ proof)

Da adesso, assumeremo che la randomicità sia data gratuitamente.

Lecture 4

[1]: 1.3, 3.1, 3.3 [2]: 3.1, 7.1 [7]: 2.1

- ➢ Computational security: Asymptotic security, negligible and polynomial functions, PPT algorithms.
- ➢ One-Way Functions and Impagliazzo's worlds.

Computational security

Statistical security comes with some inherent limitation, so we change our perspective taking into account that in the real world the adversary has a bounded power and a tiny (very small) probability of security breach. Abbiamo due approcci possibili:

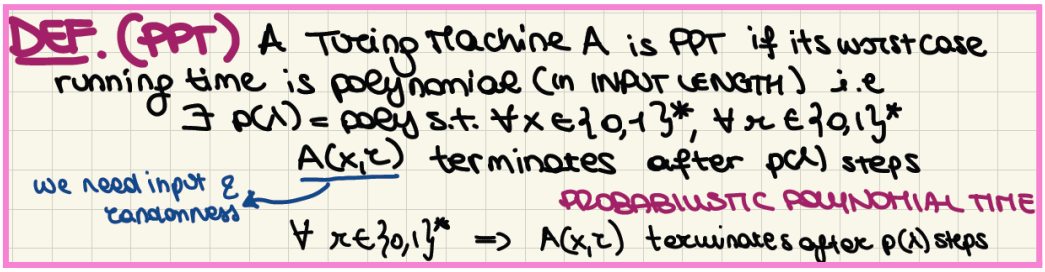
- CONCRETE SECURITY, ovvero costruire un cryptoscheme affinchè nessun adversary possa rompere lo schema in t step con probabilità maggiore di ε. Può essere importante avendo uno schema già esistente, ma non creando lo schema da principio. Dovendo noi crearlo da principio, abbandoniamo la prima opzione a favore della seconda.

- ASYMPTOTIC SECURITY, in cui il security parameter λ (con $\lambda \in \mathbb{N}$) è asintotico.

Diamo la definizione di polynomial function [Def 7 - Polynomial] e la definizione di negligible function [Def 8 - Negligible] (ossia di funzione trascurabile, funzione che tende a zero più velocemente dell'inverso di un polinomio). Intuition: pensiamo ad un algoritmo A che risolve qualche problema con probabilità p (e quindi fallisca con $1 - p = q$). A quel punto, se q è tanto grande, andiamo ad eseguire A un numero di volte molto alto per far tendere la probabilità di fallimento ad una quantità negligible. Se invece A fallisse con probabilità $1/k$ (k grandezza in input) la ripetizione non ci aiuta, ma dovremmo scegliere un k molto grande per poter avere una probabilità di fallimento molto bassa (arbitrary choice → small per essere negligible e fast per essere polynomial). Da questa discussione, possiamo ricollegarci al discorso della COMPUTATIONAL SECURITY. Per questo motivo, definiremo un approccio efficiente (rendendolo polynomial) e con errori solo con probabilità negligible (quindi la probabilità negligible è riconducibile ad avere una piccola probabilità di security breach).

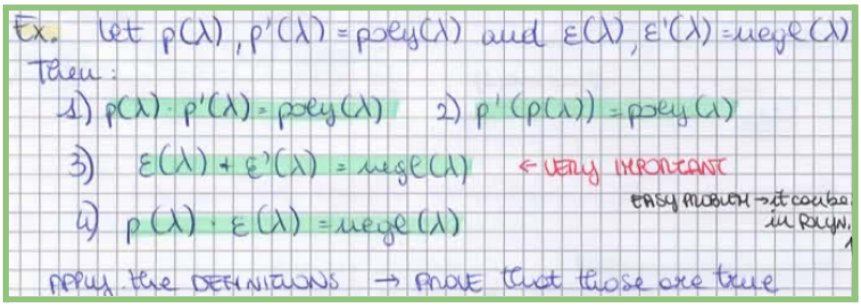
- Def 7 - Polynomial
- Def 8 - Negligible

Il PPT (Probabilistic Polynomial Time) sarà quindi il requisito per avere un algoritmo che sia efficiente [Def 9 - PPT].



- Def 9 - PPT

Exercise.



[1]: 1.3 (Complessità computazionale)

Una volta noto che un dato problema è risolvibile, potremmo pensare che non ha importanza se il tempo necessario a risolverlo è 10 o 100 secondi. Tuttavia, questa conclusione non sarebbe così ovvia se la differenza fosse 10 secondi oppure 101010 secondi! La teoria della complessità studia come le risorse necessarie a risolvere un dato problema, scalano con la "dimensione" n del problema stesso In informatica teorica, un algoritmo è detto "efficiente" se è eseguibile in un tempo che è limitato superiormente da una funzione polinomiale in n, mentre si parla di algoritmi "inefficienti" quando il tempo di esecuzione è limitato inferiormente da una funzione esponenziale di n. (...) Lo strumento universale per modellare un algoritmo (implementato in un generico calcolatore) che tenta di risolvere un dato problema, è la cosiddetta macchina di Turing. (...) La nozione di macchina di

Turing è il punto di partenza per il concetto di sicurezza computazionale. (...) Un crittosistema computazionalmente sicuro dipende da un parametro statistico di sicurezza *n* (noto a tutte le parti, attaccanti compresi). La probabilità di successo di un attacco al sistema ed il tempo necessario ad eseguire un attacco sono funzioni di tale parametro.

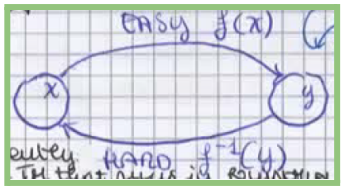
2 (Sicurezza incondizionata)

Esistono due accezioni di sicurezza per una primitiva crittografica: sicurezza incondizionata e sicurezza computazionale. Sebbene, in pratica, il secondo approccio conduca a soluzioni più efficienti, il primo approccio ha il vantaggio di non dover assumere nessuna ipotesi non dimostrata in teoria della complessità. Quando si parla di sicurezza computazionale si considerano altre due ipotesi: che le risorse computazionali a disposizione dell'avversario siano limitate e che un certo problema computazionale sia difficile, ovvero che esso richieda un tempo enorme per essere risolto dato il limite computazionale di cui al punto detto prima. Nel contesto della sicurezza incondizionata non si considerano questi due punti, ovvero si suppone che l'attaccante abbia risorse computazionali infinite.

One-Way Functions (OWF)

Sotto quali assunzioni quindi noi faremo crittografia? Noi proveremo conditional statements e baseremo la nostra crittografia su ben fondati "hard problems", ovvero problemi che non possono essere risolti in tempo polinomiale eccetto con probabilità negligible.

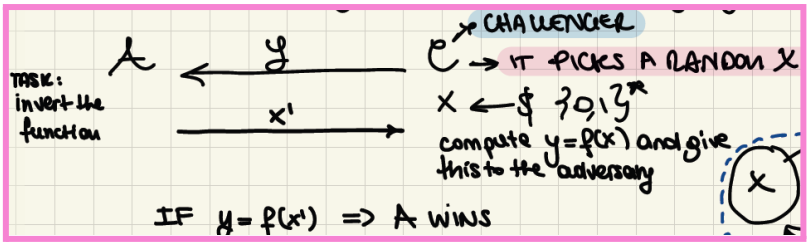
Come definiremo quindi la security? Proviamo che se il mio schema non è sicuro (ovvero che esiste un avversario efficiente che lo rompe) allora esiste uno schema sicuro che risolve il problema in polynomial time.



Definiamo una ONE-WAY FUNCTION questo semplice schema, ovvero un SECURITY GAME in cui nessun ADVERSARY può vincere [Def 10 - OWF]. Questa funzione f è:

- *easy* da computare in una direzione ($f(x)$ può essere computata efficientemente);
- *hard* da invertire.

(•) If the adversary is able to compute one pre-image, he brokes the OWF and so the OWF is not secure. Instead the OWF is secure if for all adversaries, the probability that he wins (that inverts the OWF) is negligible. Negligible in what? Negligible in the security parameter λ ($f: \{0,1\}^* \to \{0,1\}^*$}, in general n is a function of λ [$n(\lambda)$]).



- Def 10 - OWF

[1]: 3.3 (Funzioni unidirezionali e predicati estremi)

Intuitivamente una funzione unidirezionale è una funzione facile da calcolare, ma (quasi sempre) difficile da invertire. Il primo requisito è semplice da formalizzare: richiederemo che la funzione sia calcolabile in tempo polinomiale. Quanto al secondo requisito, siccome siamo interessati alla sicurezza computazionale, richiederemo che nessun avversario PPT possa invertire la funzione con probabilità non trascurabile.

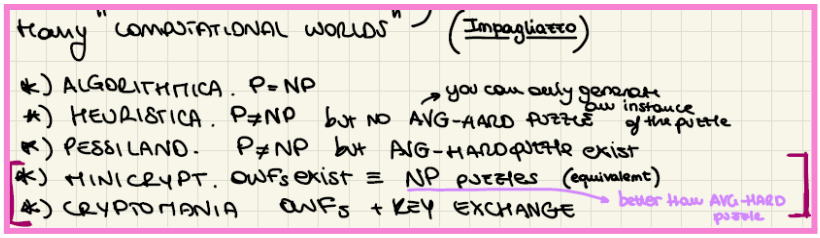
(...) In termini intuitivi la Definizione 3.4 (OWF) è equivalente al seguente esperimento mentale: per una qualche funzione f : {0, 1}∗ → {0, 1}∗ l'attaccante A conosce y = f(x) e deve calcolare un valore x

tale che f(x) = f(x) (cioè deve trovare una pre-immagine di y). Osserviamo che dato y = f(x) è sufficiente che A trovi una qualsiasi pre-immagine x che soddisfi f(x) = y, non importa che x = x.

Exercise. Prove that if $P = NP \rightarrow$ NO OWF exists (unknown the other direction: if NO OWF exists $\rightarrow P = NP$)

Impagliazzo's worlds

Queste OWF esistono? Il tutto è collegato dalla relazione tra P (classe di problemi che possono essere risolti efficientemente) ed NP. Se quindi P=NP allora non esiste nessuna ONE-WAY FUNCTION, mentre il contrario non è mai stato provato. Proprio per questo esistono diversi mondi computazionali ([1]: 3.3 (Mondi di Impagliazzo)) a seconda della relazione che viene attribuita da P ed NP. Gli unici mondi in cui la crittografia può essere applicata sono minicrypt e cryptomania (i mondi che noi andremo ad analizzare).



Lecture 5

[1]: 3.2, 5.1, 5.2 [2]: 7.2, 7.4, 3.2 [7]: 2.2, 2.3

- ➢ Computational indistinguishability and hybrid arguments.
- ➢ Definition of Pseudo-Random Generators (PRGs).
- ➢ Definition of one-time computational security for Secret Key Encryption (SKE) and construction from any PRG.

(•) In minicrypt, we assume that OWFs exist. We have two approaches:

1. base crypto on "generic" OWF f (I assume that exists f that is OWF. I don't know it, but I know that exists);
2. base crypto on "concrete" OWF f (I have a specific problem, e.g. factoring, that is OWF and base my crypto on it).

We will do both approaches, let's start with the first.
Our goal now is symmetric encryption using OWFs and in order to do that we need PRG.

Pseudorandomness

OWF è un buon crypto tool contro avversari efficienti ma ricordiamoci che il nostro PLAN iniziale era di sconfiggere Shannon. A questo obiettivo vedremo un basic tool per fare encryption basato su pseudorandomness. Pseudo poiché non è random, ma lo fa sembrare randomico all'avversario: l'avversario non può distinguere l'output dell'estrattore da una distribuzione uniforme eccetto con probabilità ε.

[1]: 3.2 Pseudorandomicità ed impredicibilità

Un generatore pseudocasuale (Pseudorandom Generator, PRG), è un algoritmo deterministico G che riceve come input un piccolo seme di vera randomicità (diciamo n bit) per produrre una lunga stringa (diciamo $l(n)$ bit) che sia "pseudocasuale". (In questo senso un PRG è un amplificatore di randomicità.) Per dare una definizione formale, useremo il concetto di indistinguibilità introdotto nel paragrafo precedente: G è pseudocasuale se la distribuzione dell'output che produce è computazionalmente indistinguibile dalla distribuzione uniforme $U_{l(n)}$ su $\{0,1\}^{l(n)}$.

Formalmente abbiamo la seguente: Definizione 3.2 (Generatore Pseudocasuale) (...)

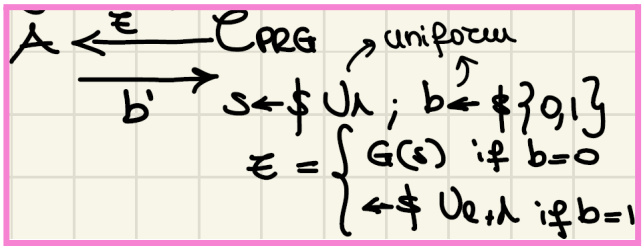
PRG

Definiamo quindi la funzione deterministica Pseudo Random Generator (PRG) $G: \{0,1\}^{\lambda} \to \{0,1\}^{\lambda+l}$ con stretch l [Def 11 - PRG]. G è una PRG se:

1. G poly-time computable; 2. $l = l(\lambda) \geq 1$; 3. $G(U_{\lambda}) \approx_c U_{\lambda+l}$.

(•) This function G take in input λ bit and it stretches in something larger. What is the security pseudorandomness? It means that the output of $G(s)$ for uniform seed s looks random. Let's do an example. Alice and Bob meet at the secret place and exchange s (that has length λ), Then they go home and obtain r that has length $\lambda + l$ (they stretch it). Now Alice and Bob know the same r. Since r looks random, then the adversary doesn't know r! Now Alice and Bob can use r to encrypt, instead of sharing directly r. If r was truly random, it would be secure; unfortunately, r is not truly random, because G is deterministic (only s is random). But r doesn't need to be random, it has only to be random from the point of view of Eve, that is a Turing Machine.

Analizziamo quindi (3), ovvero il fatto che $G(U_{\lambda})$ sia computationally close a $U_{\lambda+l}$, e definiamo la security della PRG, comparando due esperimenti. Consideriamo un real experiment ($Real$), in cui l'avversario riceve l'output della PRG da un seed uniformly random, e un random experiment ($Rand$) in cui l'avversario vede una stringa random come output della PRG. Perché facciamo questo? Perchè il nostro obiettivo è di provare che le due situazioni siano computationally close e che quindi G sembri random agli occhi dell'adversary. Definito allora il game in cui l'avversario deve distinguere in quale delle due situazioni si trova, possiamo affermare che [Def 12 (3A, 3B, 3C) - Security of PRG]:



3A. $|Pr[Real_{G.A}(\lambda) = 1] - Pr[Rand_{G.A}(\lambda) = 1]| \leq negl(\lambda)$;

3B. $|Pr[A(z) = 1\colon z = G(s);\ s \leftarrow \$\{0,1\}^{\lambda}] - Pr[A(z) = 1\colon z \leftarrow \$U_{\lambda+l}]| \leq negl(\lambda)$;

3C. $\forall PPT\ A,\ Pr[A\ wins] \leq \frac{1}{2} + negl(\lambda)$ (Nota: $\frac{1}{2}$ non è negligible, quindi potenzialmente l'avversario può vincere tirando a caso con probabilità appunto $\frac{1}{2}$).

Definition (3A) is the cleanest.

- Def 11 - PRG
- Def 12 (3A, 3B, 3C) - Security of PRG

Exercise. Prove that 3A, 3B and 3C are equivalent

Exercise (+ idea). No PRG is secure unconditionally (show something is not secure → use an unbounded adversary that breaks it → exists an unbounded A that breaks any G with non-negligible probability …) Vedi esercizio 3, Lecture 23 (2022).

(CRYPTO 2022)
Nel 2022, non si menzionano le tre definizioni di Def 12.
Nel 2022, definizione formale di computationally indistinguishable.

[Def A] Let $X = \{X_\lambda\}_{\lambda\in\mathbb{N}}$, $Y = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ be ensembles of R^n, we say that X is computationally indistinguishable from Y ($X \approx_c Y$) if $\forall PPT\ A$ then $|Pr[A(X_\lambda) = 1] - Pr[A(Y_\lambda) = 1]| \leq negl(\lambda)$ (or also equal $|Pr[A(X_\lambda) = 0] - Pr[A(Y_\lambda) = 0]| \leq negl(\lambda)$).

- Def A - Computationally indistinguishable

Definition of one-time computational security for SKE and construction from any PRG

Ora che abbiamo definito la PRG, questo sarà il nostro strumento per battere Shannon. Come? Unificando PRG con lo schema definito precedentemente (OWF).
PLAN:

1. OWFs ⇒ PRGs with $l(\lambda) = poly(\lambda)$
2. PRGs ⇒ Secure encryption.

2. PRGs ⇒ Secure encryption.
Iniziamo dal punto 2. È necessario prima di tutto definire la secure encryption (computationally secure) per poter sconfiggere Shannon (Shannon is inherently inefficient, vedi fine Lecture 1 e Thm 3). Come definire la secure encryption? Come avevamo detto (Shannon), abbiamo secure encryption quando il ciphertext non rivela nulla del plaintext, anche se l'avversario non è bounded → possiamo tradurre quindi la definizione di Shannon nel computationally world, dove l'avversario è computazionalmente bounded.
(•) There are several definitions of secure encryption. What should be hard?

1. hard to recover message from ciphertext;
2. hard to recover the key as well
   - These are very weak.
   - Not good! Why?
     For instance, take $Enc(k, m) = m$ with k random → so the adversary cannot recover k but m yes.
     What about $Enc(k, m||b) = b||k \oplus m$ (k and m of n bits, b is one bit)? It's hard to recover the message, but a part of it is clear. We want to hide the message, but also partial information about this.

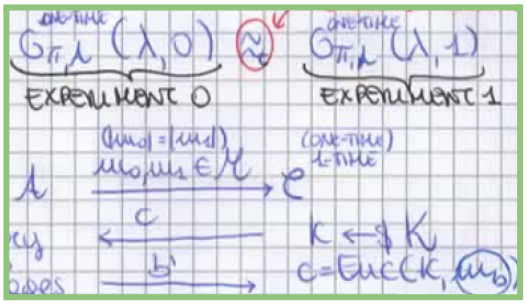
So we need to have definitions that imply *1* and *2* but are better. We will also see that "no partial info" is implied.

[1]: 5.1 (Nozioni di sicurezza per cifrari simmetrici)
Come visto, la conseguenza fondamentale del teorema di Shannon è che la sicurezza incondizionata richiede di usare un'unica chiave per cifrare un singolo messaggio; inoltre tale chiave deve essere almeno tanto lunga quanto il messaggio da cifrare. Siccome ciò non è molto efficiente, vorremmo trovare un modo per cifrare un numero arbitrario di messaggi usando la stessa chiave k, mantenendo

allo stesso tempo un livello accettabile di sicurezza. Inoltre, vorremmo che la chiave fosse possibilmente corta e che si possa utilizzarla per cifrare messaggi arbitrariamente lunghi. Come anticipato, l'idea è quella di limitare il potere computazionale della Regina, passando dal concetto di sicurezza incondizionata a quello della sicurezza computazionale. Considereremo quindi solo attacchi eseguibili in tempo polinomiale (PPT). Prima di procedere oltre, sottolineiamo due limitazioni inerenti ogni cifrario che usa una chiave a lunghezza n molto più corta della lunghezza dei messaggi che cifra: (i) in tempo 2^n l'attaccante può enumerare tutte le possibili chiavi ed usarle per decifrare il crittotesto in 2^n modi diversi, uno dei quali è quello corretto e (ii) l'avversario può indovinare la chiave con probabilità 2^{-n}. Quello descritto non è altro che l'attacco a forza bruta. Comunque, già quando n = 128, nessuna strategia è allarmante per Alice ed il Bianconiglio: anche disponendo del computer più potente della Terra, Alice ed il Bianconiglio sarebbero già morti di vecchiaia prima che la Regina termini l'enumerazione di tutte le 2128 chiavi. Inoltre la probabilità che la Regina indovini la chiave è molto più bassa della probabilità che Alice sia colpita da una meteora!

Silvio Micali ci fornisce una definizione equivalente e semplificata di secure encryption, rispetto alla precedente basata su una random key. Diciamo che una primitiva Π è ONE TIME SECURITY [Def 13 - OTS], se l'avversario non è in grado di distinguere quale messaggio (tra m_0 e m_1) sia stato encrypted dal challenger, all'interno del game in figura (Cosa ci ricorda? Definizione di PRG, Def 11). OTS implies the property "hard to compute the key", because if A could compute the key then he could distinguish between the two encryptions.

(•) If I can recover the key, I can also decrypt and if I can decrypt then I know what is m_0 and m_1. The difference of the two probabilities is equal to 1 and so it is not negligible (contradiction). For the similar argument, the definition implies that it's hard to recover the key and also the first bit of the message. The adversary can choose m_0 and m_1 that differs only for the first bit, then receives c and he can distinguish them (contradiction).



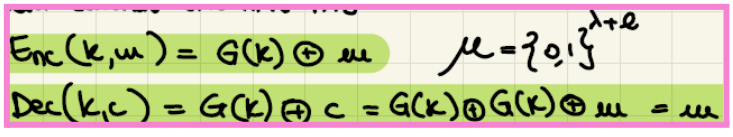
- Def 13 - OTS

Exercise. Write this definition (similar to 3A) using what we used in 3C definition

Exercise (+ idea). Above definition implies $Pr[A(c) = k: k \leftarrow \$\mathcal{K}; c = Enc(k.m)] \leq negl(\lambda)$ $\forall PPT\ A, \forall m \epsilon \mathcal{M}$. If this wasn't true, exists A could distinguish m from m' and so the above definition doesn't hold.

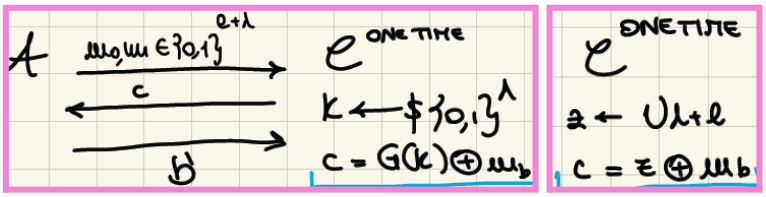
Exercise (similar to exam). Prove that perfect secrecy implies computational OTS.

[Construction] Costruiamo quindi la nostra primitiva Π sulla nostra PRG $G: \{0,1\}^\lambda \rightarrow \{0,1\}^{\lambda+l}$.



Assumendo che G sia una secure PRG, allora proviamo mediante il teorema [Thm 7 - PRG⇒Secure encryption] che Π è One Time Security (OTS). How to prove that something is secure? We must 1. assume that it's not secure and 2. contradict the assumption.

We start the prove, applying our construction with G to the game of the OTS definition (Def 13). In this game $G^{One-time}_{\Pi, A}(\lambda, b)$, A is bounded because of PRG hypothesis.



Per provare che i due game $G^{One-time}_{\Pi, A}(\lambda, 0)$ e $G^{One-time}_{\Pi, A}(\lambda, 1)$ sono computationally close, utilizziamo la tecnica dell'hybrid argument (ovvero dei game intermedi corrispondenti a esperimenti mentali): so we use an intermediate game $Hyb(\lambda, b)$. In this game A can be unbounded, but we care about bounded adversary that can distinguish between the two worlds. Observation. Hybrid argument technique works for at most $poly(\lambda)$ experiments. Procediamo quindi definendo due lemmi.

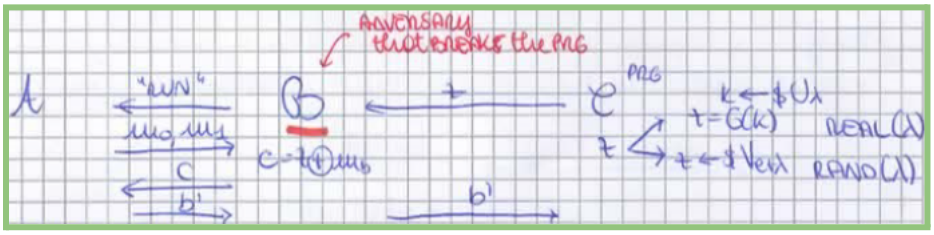
- [Lemma 1] $G^{One-time}_{\Pi, A}(\lambda, b) \approx_c Hyb(\lambda, b),\ \forall b \epsilon \{0, 1\}$;

We prove the first lemma by reduction. Assume lemma is not true…

→ then $\exists PPT\ A\ s.t\ |Pr[G^{One-time}_{\Pi, A}(\lambda, b) = 1] - Pr[Hyb(\lambda, b) = 1]| \geq 1/poly(\lambda)$ for some b

→ and so $\exists PPT\ B\ s.t\ |Pr[Real_{G,B}(\lambda) = 1] - Pr[Rand_{G,B}(\lambda) = 1]| \geq 1/poly(\lambda)$.

If A exists, then exists B but B contradicts the assumption that G is a PRG: so A not exists. B is called a reduction (we are reducing A to B) and so we proceed to build B.



How does reduction work? Step 1. Describe the reduction; Step 2. Prove that the reduction is correct (analyzing it). By inspection:

$Pr[Real_{G,B}(\lambda) = 1] = Pr[G^{One-time}_{\Pi, A}(\lambda, b) = 1]$

$Pr[Rand_{G,B}(\lambda) = 1] = Pr[Hyb(\lambda, b) = 1]$

The reduction is perfect, because when the reduction receives z, which is computed by the PRG, the view of A is perfectly identical to the view of A in the real game when the challenger encrypts B. By assumption:

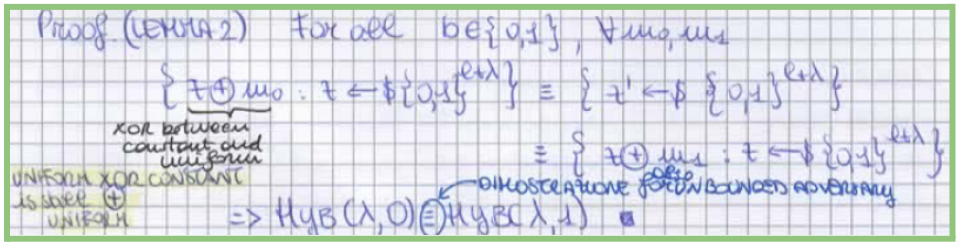
$|Pr[Real_{G,B}(\lambda) = 1] - Pr[Rand_{G,B}(\lambda) = 1]| =$

$|Pr[G^{One-time}_{\Pi, A}(\lambda, b) = 1] - Pr[Hyb(\lambda, b) = 1| \geq 1/poly(\lambda)$.

Because $|Pr[Real_{G,B}(\lambda) = 1] - Pr[Rand_{G,B}(\lambda) = 1]| \geq 1/poly(\lambda)$, the definition of PRG is broken (Def 12 (3A)). B has broken the PRG, so this proved the lemma.

- [Lemma 2] $Hyb(\lambda, 0) \equiv Hyb(\lambda, 1)$ (identical distribution, same probability for A to output in both cases, $b = 0$ or $b = 1$, even if the adversary is unbounded).

Now we must prove the second lemma.

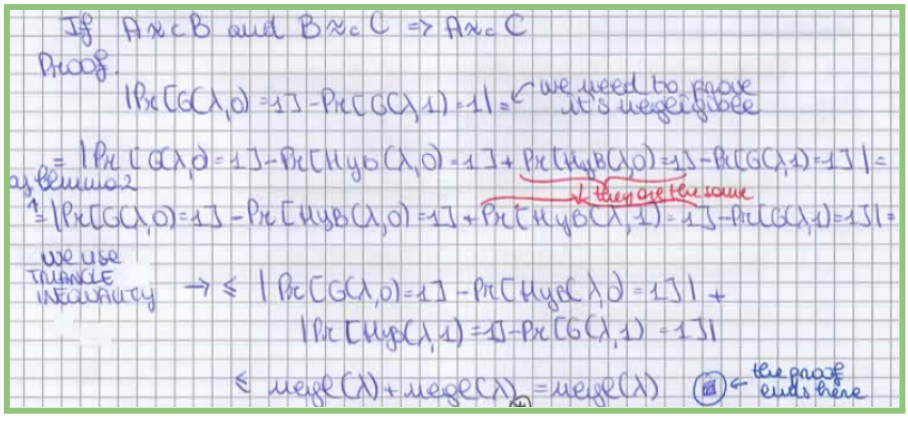


This proof follows from One Time Pad: the distribution of the ciphertext is independent from b : $z \oplus$ *something constant* is still uniform, so for each message the distribution of $z \oplus m$ is uniformly random. The probability of distinguishing the two experiments is zero: the two experiments are the same.

Quindi proviamo il teorema: $G^{One-time}_{\Pi, A}(\lambda, 0) \approx_c Hyb(\lambda, 0) \equiv Hyb(\lambda, 1) \approx_c G^{One-time}_{\Pi, A}(\lambda, 1)$.

- (Construction +) Thm 7 - PRG⇒Secure encryption (+ proof) + Lemma 1 (+ proof) + Lemma 2 (+ proof)

Exercise (+ solution). $A \approx_c B$ and $B \approx_c C \rightarrow A \approx_c C$



[1] 3.1 Indistinguibilità ed argomento ibrido

Un modo naturale per esprimere formalmente il concetto di pseudorandomicità è attraverso l'uso del concetto di indistinguibilità. Informalmente diremo che una sequenza di valori è pseudocasuale se è impossibile (o comunque difficile) distinguerla da una sequenza di valori veramente casuale. Come vedremo il paradigma dell'indistinguibilità è usato in svariati contesti in crittografia. (...)

Definizione 3.1 (Indistinguibilità)

Definizione 3.1 (Indistinguibilità). Sia $n \in \mathbb{N}$ un parametro statistico di sicurezza ed indichiamo con $X = \{X_n\}_{n\in\mathbb{N}}$ ed $Y = \{Y_n\}_{n\in\mathbb{N}}$ due insiemi di distribuzioni di probabilità efficientemente campionabili. Diremo che:

(i) X ed Y sono *identicamente distribuiti*, indicato con $X \stackrel{d}{\equiv} Y$, se hanno esattamente la stessa distribuzione;

(ii) X ed Y sono *computazionalmente* (t,ϵ)-indistinguibili, indicato con $X \stackrel{c}{\approx} Y$, se per ogni attaccante PPT $\mathscr{D}$ eseguibile in tempo t e per ogni n sufficientemente grande, abbiamo che

$$|\mathbb{P}[\mathscr{D}(X_n) = 1] - \mathbb{P}[\mathscr{D}(Y_n) = 1]| \leq \epsilon(n),$$

dove la probabilità è presa sulla randomicità di $\mathscr{D}$ e di X_n ed Y_n;

(iii) X ed Y sono *statisticamente* ϵ-indistinguibili, indicato con $X \stackrel{s}{\approx} Y$, se per ogni n sufficientemente grande, abbiamo che

$$\Delta(X_n, Y_n) \leq \epsilon(n),$$

dove $\Delta(\cdot)$ è l'operatore di distanza statistica tra distribuzioni (cf. Definizione A.3). ■

Osserviamo che l'attaccante $\mathscr{D}$ al punto (ii) della Definizione 3.1 restituisce un valore in $\{0,1\}$ come tentativo di distinguere X_n da Y_n. Notare inoltre che l'indistinguibilità statistica non fa alcuna ipotesi sul potere computazionale dell'attaccante. (In effetti essa è usata nel contesto della sicurezza incondizionata.) Non è difficile mostrare (come l'intuizione suggerisce) che l'indistiguibilità statistica implica quella computazionale (cf. Esercizio 3.2); tuttavia la direzione opposta non è verificata (ma una dimostrazione di questo fatto è più complessa [Gol01, Proposizione 3.2.3]).

Prima di procedere oltre (e vedere come applicare il concetto d'indistinguibilità per generare sequenze binarie pseudocasuali) introduciamo una tecnica molto usata in crittografia detta *l'argomento ibrido*. Tale tecnica permette di dimostrare che due insiemi di distribuzioni sono indistinguibili definendo una serie di *distribuzioni ibride intermedie* e mostrando che gli ibridi consecutivi sono indistinguibili.

Teorema 3.1 (Argomento ibrido)

Teorema 3.1 (Argomento ibrido). *Sia $\ell = \ell(n)$ un polinomio in n, ovvero $\ell = \mathrm{poly}(n)$. Se $X^{(0)} \stackrel{c}{\approx} X^{(1)}$, $X^{(1)} \stackrel{c}{\approx} X^{(2)}, \ldots, X^{(\ell-1)} \stackrel{c}{\approx} X^{(\ell)}$, allora $X^{(0)} \stackrel{c}{\approx} X^{(\ell)}$. Più precisamente, se per ogni indice $i = 0, \ldots, \ell - 1$ gli insiemi di distribuzioni $X^{(i)}$ ed $X^{(i+1)}$ sono $(t, \epsilon/\ell)$-indistinguibili, allora $X^{(0)}$ ed $X^{(\ell)}$ sono (t,ϵ)-indistinguibili.*

Lecture 6

[1]: 3.3, 3.4 [2]: 3.3, 7.3 [7]: 2.4

- ➢ Definition of hard-core predicates.
- ➢ Goldreich-Levin theorem (statement).
- ➢ Proof that One-Way Permutations (OWPs) imply PRGs with one-bit stretch.
- ➢ Proof that one-bit stretch is sufficient to obtain arbitrary polynomial stretch.

1. OWFs ⇒ PRGs with $l(\lambda) = poly(\lambda)$

Ora passiamo alla prima parte del PLAN e per dimostrare questo utilizziamo il seguente approccio:

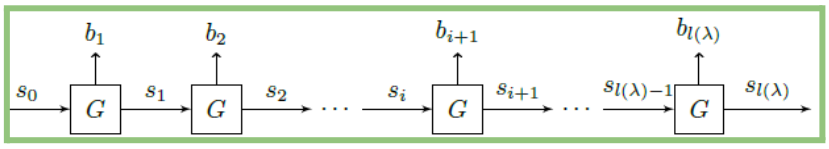
a. PRGs with $l(\lambda) = 1$ ⇒ PRGs with $l(\lambda) = poly(\lambda)$

b. OWPs (OWFs) ⇒ PRGs with $l(\lambda) = 1$

Proof that one-bit stretch is sufficient to obtain arbitrary polynomial stretch

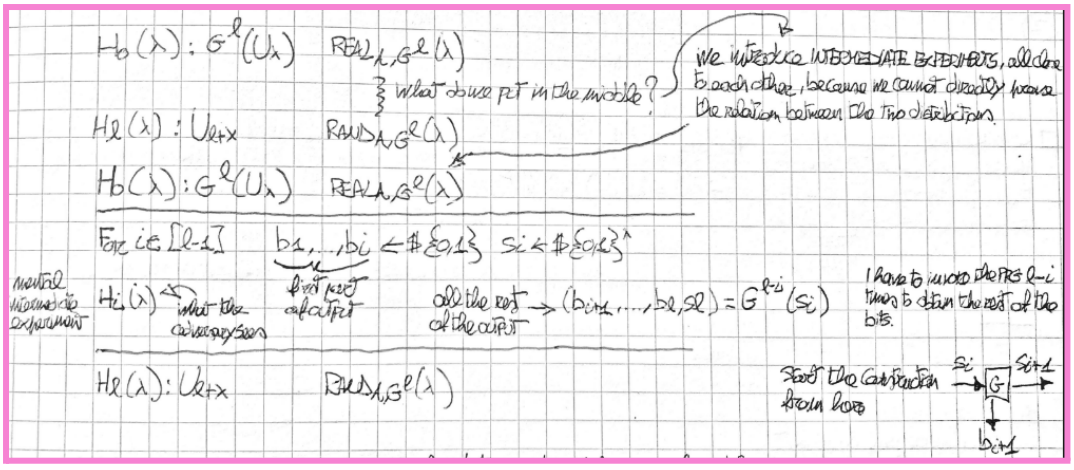
a. PRGs with $l(\lambda) = 1 \Rightarrow$ PRGs with $l(\lambda) = poly(\lambda)$

Partiamo dal punto ***a***, questo viene dimostrato mediante un teorema [Thm 8 - If there exists PRG $G: \{0,1\}^{\lambda} \to \{0,1\}^{\lambda+1}$ then there exists PRG $G^{l}: \{0,1\}^{\lambda} \to \{0,1\}^{\lambda+l}$, $\forall l(\lambda) = poly(\lambda)$]. La dimostrazione utilizza la seguente costruzione [Construction]:



- $G^{l}: \{0,1\}^{\lambda} \to \{0,1\}^{\lambda+l}$
- $s_o \leftarrow \$\{0,1\}^{\lambda}$
- $\forall i \epsilon [l]$: $(s_i, b_i) = G(s_{i-1})$
- Output: $(b_1, \dots, b_l, s_l) \epsilon \{0,1\}^{\lambda+l}$

For PRG property, PRG outputs something pseudorandom but only when the seed is random: b_1 and s_1 are pseudo random because s_0 is random, but what about the other outputs? We need to prove that they are pseudorandom even if it does not come directly from the definition of PRG. How do we prove it? We can prove it by reduction. The output has to be computationally indistinguishable from a uniformly random string of $\lambda + l$ bits. Intuition: we will use the technique of hybrid argument and we will not go directly from the real experiment to the rand experiment, introducing intermediate experiments, all close to each other.



Now in order to prove that the extremes are computationally close ($G^{l}(U_{\lambda}) \approx_c U_{\lambda+l}$), we define a lemma [Lemma]: for all $i \epsilon [l]$: $H_i(\lambda) \approx_c H_{i+1}(\lambda)$.

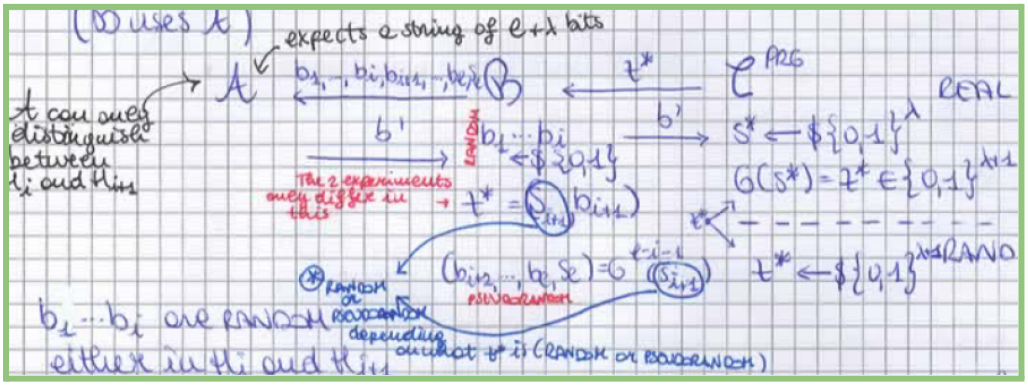
Now we can do the reduction: assume lemma is not true

$\rightarrow$ then $\exists PPT\ A\ s.t\ |Pr[H_i(\lambda) = 1] - Pr[H_{i+1}(\lambda) = 1]| \geq 1/poly(\lambda)$

(i.e. A can distinguish between $H_i(\lambda)$ and $H_{i+1}(\lambda)$)

$\rightarrow$ then $\exists PPT\ B\ s.t\ |Pr[Real_{G,B}(\lambda) = 1] - Pr[Rand_{G,B}(\lambda) = 1]| \geq 1/poly(\lambda)$

(i.e B breaks G). So we can proceed to construct B.



If we analyze $H_i(\lambda)$ and $H_{i+1}(\lambda)$:

- the first i bits $(b_1, \dots, b_i)$ are identical for the two experiments
- the difference is $z^* = (b_{i+1}, s_{i+1})$, in particular…
  - $H_i(\lambda) \to b_{i+1}$ and in s_{i+1} are pseudorandom ($z^* = (b_{i+1}, s_{i+1}) = G^{l-i}(s_{i+1})$)
  - $H_{i+1}(\lambda) \to b_{i+1}$ and in s_{i+1} are random ($z^* = (b_{i+1}, s_{i+1}) = U_{\lambda+1}$)
- the others are pseudorandom, $(b_{i+2}, \dots, b_l, s_l) = G^{l-i-1}(s_{i+1})$

By inspection: - if z^* from $Real$: view of A like H_i ; - if z^* from $Rand$: view of A like H_{i+1} .

(•) - $Pr[B(z^*) = 1: z = G(s);\ s \leftarrow \$U_\lambda] =$

$= Pr[A(b_1, \dots, b_l, s_l) = 1, (b_1, \dots, b_l, s_l) \leftarrow \$H_i(\lambda)]$

- $Pr[B(z^*) = 1: z \leftarrow \$U_{\lambda+1}] =$

$= Pr[A(b_1, \dots, b_l, s_l) = 1, (b_1, \dots, b_l, s_l) \leftarrow \$H_{i+1}(\lambda)]$ (contradiction!)

B using A can distinguish the two distributions and so it breaks PRG. This proves the lemma and also the theorem.

- (Construction +) Thm 8 - If there exists PRG $G: \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ then there exists PRG $G^l: \{0,1\}^\lambda \to \{0,1\}^{\lambda+l}$, $\forall l(\lambda) = poly(\lambda)$ (+ proof) + Lemma (+ proof)

(CRYPTO 2022)

Nel 2022, la dimostrazione del Thm 8 utilizza un lemma differente, ma concettualmente uguale; prosegue poi come nel 2021.

[Lemma] If $\{X_\lambda\} \approx_c \{Y_\lambda\}$ and $\{Y_\lambda\} \approx_c \{Z_\lambda\}$ then $\{X_\lambda\} \approx_c \{Z_\lambda\}$.

Proof. $\{X_\lambda\}$ and $\{Y_\lambda\}$ can be distinguished with probability $< \varepsilon$, that is negligible; $\{Y_\lambda\}$ and $\{Z_\lambda\}$ can be distinguished with probability $< \epsilon$, that is negligible; so I can apply triangular inequality and say that $\{X_\lambda\}$ and $\{Z_\lambda\}$ can be distinguished with probability $\epsilon + \epsilon'$, that is still negligible.

Hybrid argument: if I want to prove that $X \approx_c Y$: $X \equiv H_0 \approx_c H_1 \approx_c H_2 \approx_c \dots \approx_c H_{n-1} \approx_c H_n \equiv Y$ so long as $n = poly(\lambda)$ (n must be polynomial because $poly(\lambda) * negl(\lambda) = negl(\lambda)$).

So I can apply the lemma multiple times to prove the theorem.

Normally, I would like to prove that G^l is secure by reduction, proving that if G^l is not secure then G is not secure: this reduction is messy because I call G many times and with a

pseudorandom key. To reduce the number of G invocations, I compute G $l-1$ times and I consider S_i and b_i after each computation uniform: the intermediate arguments consider i executions of G as truly random and $l-i$ as pseudorandom (vedi figura sopra). So I prove $G^l(U_\lambda) \approx_c U_{\lambda+l}$, proving that $H_i(\lambda) \approx_c H_{i+1}(\lambda)$.

Proof that OWPs imply PRGs with one-bit stretch

b. OWPs (OWFs) ⇒ PRGs with $l(\lambda) = 1$

Una volta provato questo passiamo alla seconda parte dell'approccio ***b***. Prima di passare alla dimostrazione, definiamo una particolare funzione h polynomial time computable hard-core predicate [Def 14 - Hard-Core Predicate] per una funzione f: a polynomial time computable function $h: \{0,1\}^\lambda \to \{0,1\}$ is a hard-core predicate for $f: \{0,1\}^\lambda \to \{0,1\}^\lambda$ if $(f(U_\lambda), h(U_\lambda)) \approx_c (f(U_\lambda), U_1)$. [(•) if $(f(x), h(x)) \approx_c (f(x), b),\ b \leftarrow \$\{0,1\},\ x \leftarrow \$\{0,1\}^n$]

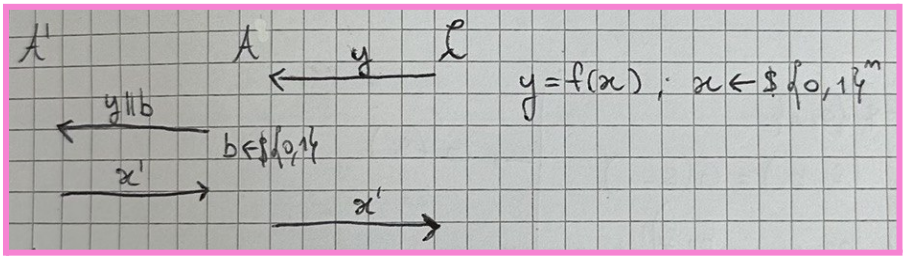
(•) Example. For $f'(x) = x[1]||f(x)$ the first bit of x is not hard-core. $h(x)$ is hard core if extracts an hard-core bit from x such that we can't distinguish between $(f(x), h(x))$ and $(f(x), b)$ where b is uniform.

Intuition: think of $h(x)$ as what is hard to compute about x given $f(x)$ or what is hard to compute about the input of f. But what is hard to compute about the input of f? Hardness of computing → undistinguishability from uniform.

- Def 14 - Hard-Core Predicate

Esiste una singola h hard-core per ogni f OWF? No.

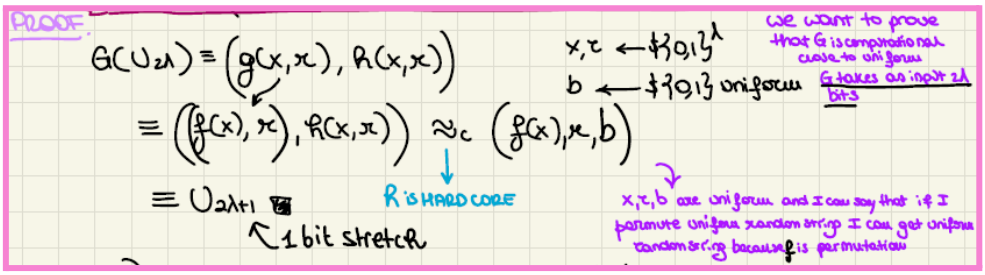
Exercise (+ solution). Prove that there is no single h that is hard-core for all OWFs. (•) There is no $h: \{0,1\}^n \to \{0,1\}$ that is hard-core for every $f: \{0,1\}^n \to \{0,1\}^n$ (for each h there is an f for which h is not hard-core). (Proof) Fix every h. Assume that h works with every f. We can always find a *BAD* f for which h doesn't work. In fact. if $f(x)$ is OWF, I can consider $f'(x) = f(x)||h(x)$ and then f' is a OWF and $h(x)$ is not hard-core for f'. So we can prove by reduction: I assume that if there is an adversary A' that breaks f', then there is an adversary A that breaks f.



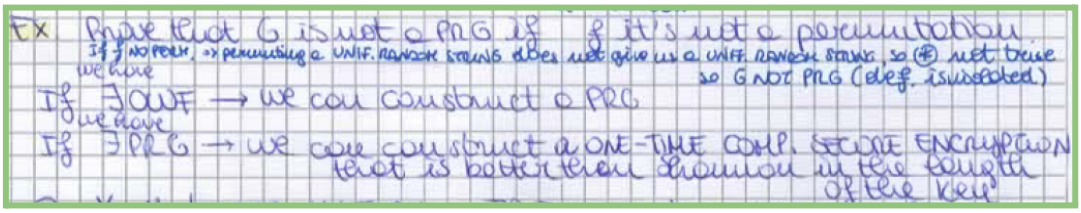
A sends $y = f(x)$ to A' concatenating to it a random bit b. The probability that A' wins if b is correct is $\geq 1/poly(1/\lambda)$. Then the $Pr[A\ wins] \geq 1/(2 * poly(1/\lambda))$.

Although I can't have an h that is hard-core for each f, I am sure that, for each f, it exists an h. [Thm 9 - Goldreich-Levin] Let f be a OWF and define $g(x,r) = (f(x), r)$ where $g: \{0,1\}^{2\lambda} \to \{0,1\}^{2\lambda}$ and $r \epsilon \{0,1\}^\lambda$. Then g is a OWF (if f is a OWF) and $h(x,r)$ equal to $h(x,r) =< x, r >= \Sigma_{i=1}^{\lambda} x_i r_i$ $((mod\ 2)$ if we want one bit as output) is hard-core for g.

Exercise (+ idea). Assume f is a OWF, prove that $g(x,r) = (f(x),r)$ is OWF. Prove by reduction that g is a OWF: if g is not OWF, then exist an adversary that breaks f.
In aggiunta, il corollario di questo teorema [Corollario - Let f be a OWP and g, h be as in the theorem, then $G(s) = (g(s), h(s))$ is a PRG with $l(\lambda) = 1$] dimostra che: presa f OWP, e g OWF e h hard-core per g (g e h definite come in Def 14), allora G costruita su g e h, $G(s) = (g(s), h(s))$, è una PRG con stretch $l(\lambda) = 1$.



Exercise (+ idea). Prove that G is not a PRG if f it's not a permutation.



- Thm 9 - Goldreich-Levin + Corollario - Let f be a OWP and g, h be as in the theorem, then $G(s) = (g(s), h(s))$ is a PRG with $l(\lambda) = 1$ (+ proof)

(CRYPTO 2022)
Nel 2022, enunciato come teorema [Thm A] il fatto che per ogni OWF f esiste una h.

- Thm A - Every OWF $f: \{0,1\}^n \to \{0,1\}^n$ has some HARD-CORE bits

Nel 2022, enunciato come teorema [Thm B] il Corollario di Goldreich-Levin.
Proof. By hard-core definition (Def 14), $G(U_n) = f(U_n)||h(U_n) \approx_c f(U_n)||U_1 \equiv U_{n+1}$.

- Thm B - Let $f: \{0,1\}^n \to \{0,1\}^n$ be a OWP. Then $G(s) = f(s)||h(s)$ is a PRG with $l = 1$ (+ proof)

[1]: 3.3 (Funzioni unidirezionali e predicati estremi)
Predicati estremi. Sia f(·) una OWF, sappiamo che è difficile determinare x a partire da f(x).
Ma quanto è difficile calcolare un particolare bit di x? Potremmo essere portati a pensare che siccome f(x) non è invertibile, non è possibile ricavare alcuna informazione su x in tempo polinomiale. Questo non è affatto vero. Esistono, infatti, esempi di funzioni che non sono invertibili in tempo polinomiale, eppure rivelano parecchia informazione su x. Ad esempio è facile vedere che la funzione g(x1, x2) = (x1, f(x2)) è una OWF, seppur evidentemente riveli metà del suo input! Si definisce allora il concetto di "predicato estremo" o anche "bit estremo" (predicato hard-core o anche bit hard-core) per una funzione unidirezionale, come segue: Definizione 3.5 (Predicati estremi) (...)
Una funzione non deve essere necessariamente unidirezionale per possedere un predicato estremo. D'altra parte in crittografia siamo interessati a costruire predicati estremi per funzioni unidirezionali. Il seguente importante teorema, dovuto a Goldreich e Levin, mostra che in effetti ciò è possibile per ogni funzione unidirezionale. Teorema 3.6 (Predicati estremi di Goldreich e Levin) (...)
[1]: 3.4 (Alcune costruzioni di PRG)
Costruzioni teoriche. Mostriamo prima come sia possibile costruire un PRG che amplifica il suo input di un solo bit, i.e. tale che $l(n) = n + 1$. Come vedremo un PRG di questo tipo può essere usato per

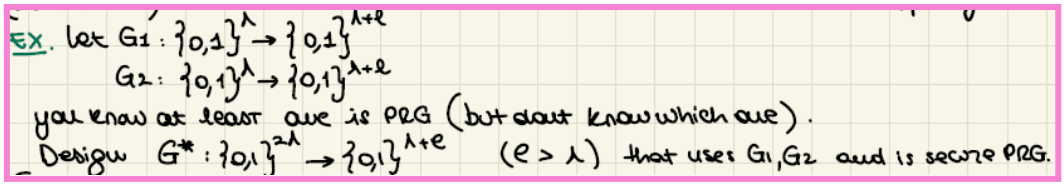
ottenere un'amplificazione di randomicità polinomiale. Håstad, Impagliazzo, Levin e Luby hanno mostrato come costruire un PRG da una qualsiasi funzione unidirezionale. Tuttavia la prova di questo fatto fondamentale è molto complessa ed esula dagli scopi del testo. Ci limiteremo invece a mostrare come sia possibile costruire un PRG da ogni permutazione unidirezionale (One-Way Permutation, OWP). Definizione 3.6 (Permutazione unidirezionale) (...)

Lecture 7

[1]: 5.1, 5.2 [2]: 7.5, 3.4, 3.5 [7]: 3.1

- Definition of Pseudorandom Functions (PRFs).
- Constructing PRFs from PRGs: The GGM construction.
- Definition of chosen-plaintext attack (CPA) secure SKE and construction from any PRF family.

Exercise (similar to exam).

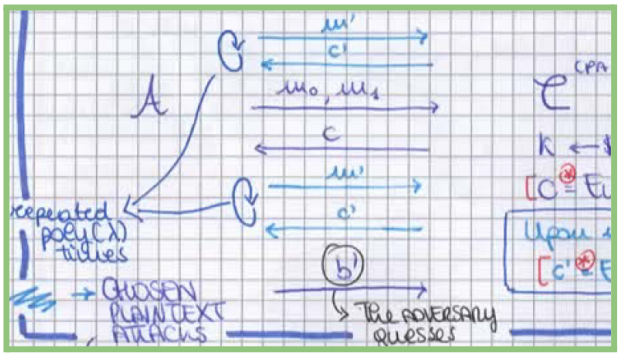


CPA secure SKE

Fino ad ora abbiamo realizzato la corrispondenza tra OWF e OTS (One Time Security) encryption scheme mediante la nozione di PRG (OWF ⇒ PRG, PRG ⇒OTS).
(•) OWF ⇒ PRG ($l = 1$) ⇒ PRG ($l = poly(\lambda)$) ⇒ SKE (One Time Security and $|k| << |m|$)
Recall the construction of Π using G as PRG: $Enc(k, m) = G(k) \oplus m = c$.
What if A knows a pair m', c' such that $c' = G(k) + m'$? A could know some encryption with k: $m_2 = c_1 \oplus c_2 \oplus m_1$ (chosen plaintext attack).
PLAN: arrivare a più encryption con la stessa chiave.
(•) I want SKE CPA secure: Alice and Bob go to the secure place, share a short secure key and use it to encrypt as many messages as they like.

CPA security

Definiamo per prima cosa una primitiva Π Chosen Plaintext (the adversary can choose the plaintext) Attack secure [Def 15 - CPA-security]: we say $\Pi = (Enc, Dec)$ is CPA secure if $Game^{CPA}_{\Pi,A}(\lambda, 0) \approx_c Game^{CPA}_{\Pi,A}(\lambda, 1)$. Questa primitiva ci restituisce un numero illimitato di ciphertext con short key/seed. La differenza con la definizione di One Time Security (Def 13) è che, sia prima che dopo aver scelto i messaggi m_0 e m_1, l'avversario può sceglierne altri a suo piacere e richiedere di vedere la corrispondente cifratura sperando di imparare qualcosa che l'aiuti poi ad indovinare il bit b' ([1]: 5.1 (Sicurezza contro attacchi a messaggio scelto)), all'interno del game in figura.



- Def 15 - CPA-security

Problema: no deterministic encryption can be CPA-secure.
Nessuna encryption deterministica ($c = Enc(k, m)$) potrà essere mai CPA secure: questo è vero, perché nessuno vieta all'adversary di chiedere al/ challenger di criptare m_0 o m_1 (o entrambi) prima di iniziare la sfida del game. La soluzione per ottenere CPA secure encryption consiste nel ritornare differenti ciphertexts per lo stesso messaggio, o anche meglio che sembrino random.
Cosa succederebbe se l'encryption fosse randomizzata ($c = \$Enc(k, m)$)? Allora m_0 e m_1 potranno essere criptati $poly(\lambda)$ volte e tutte queste encryption saranno molto differenti dal seed del challenger con alta probabilità, ovvero esisterà solo una negligible probabilità che A vinca.

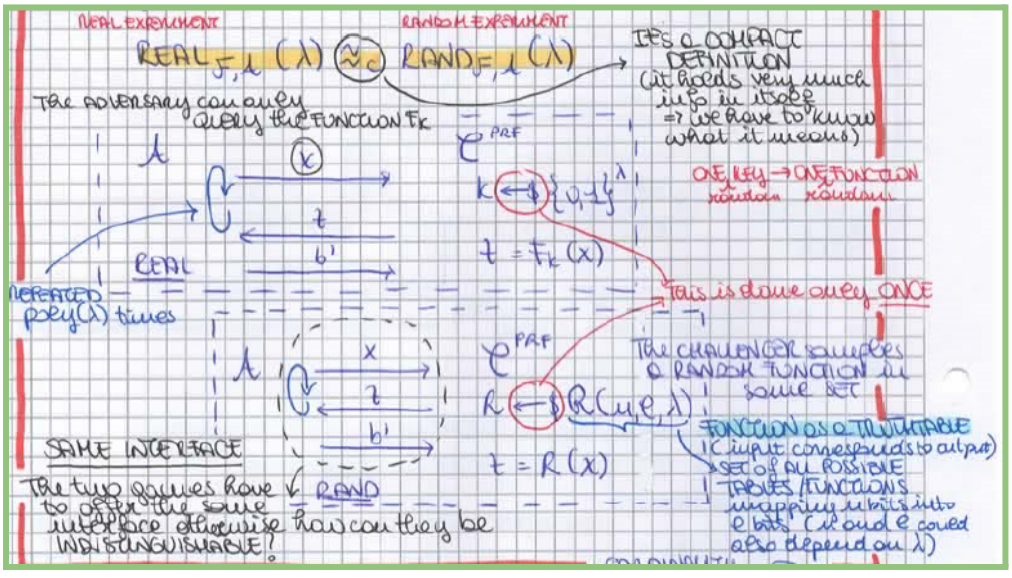
Ripartendo dall'assunzione che "no deterministc encryption can be CPA-secure", il teorema [Thm 10 - OWFs⇒CPA-secure SKE] prova che ad ogni OWF corrisponde un CPA-secure SKE (Symmetric Key Encryption).

- Thm 10 - OWFs⇒CPA-secure SKE

(CRYPTO 2022)
Nel 2022, Thm 10 non enunciato come teorema.

PRF

Per provare il teorema è necessario introdurre un nuovo tool: le pseudorandom function (PRF). Definiamo PRF una famiglia di funzioni $\mathcal{F} = \{F_k : \{0.1\}^n \to \{0.1\}^l\}_{k \in \{0.1\}^\lambda}$ se i due game real experiment (che utilizza le funzioni di $\mathcal{F}$) e rand experiment (che si basa su una funzione che lavora come una truth table, mappando n bits in l bits, con n e l anche dipendenti da λ) risultano essere computazionalmente indistinguibili [Def 16 - PRF]. F_k è dunque efficiente e pseudorandom.



(Alternative rand experiment) Il challenger non deve per forza conoscere tutta la tabella perchè l'adversary è polynomially bounded. Inoltre, la tabella si può fissarla all'inizio oppure

computare al volo (Nota: per questo motivo esiste anche un'alternativa di rand dove tabella fatta al volo e vi è una corrispondenza univoca tra x e z).

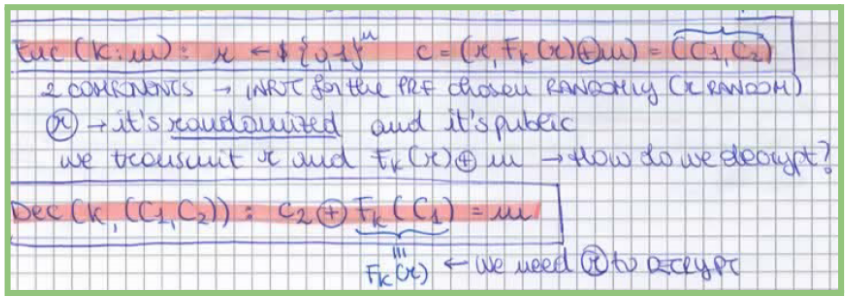
- Def 16 - PRF

(•) In minicrypt, OWF⇒PRG⇒PRF⇒CPA SKE (here not only the key is shorter, but also we are gonna be able to encrypt arbitrary many messages with the same key).
Procediamo dunque alla dimostrazione del teorema OWFs⇒CPA-secure SKE (Thm 10) e la dividiamo in due come abbiamo fatto in precedenza per le PRGs.
PLAN:

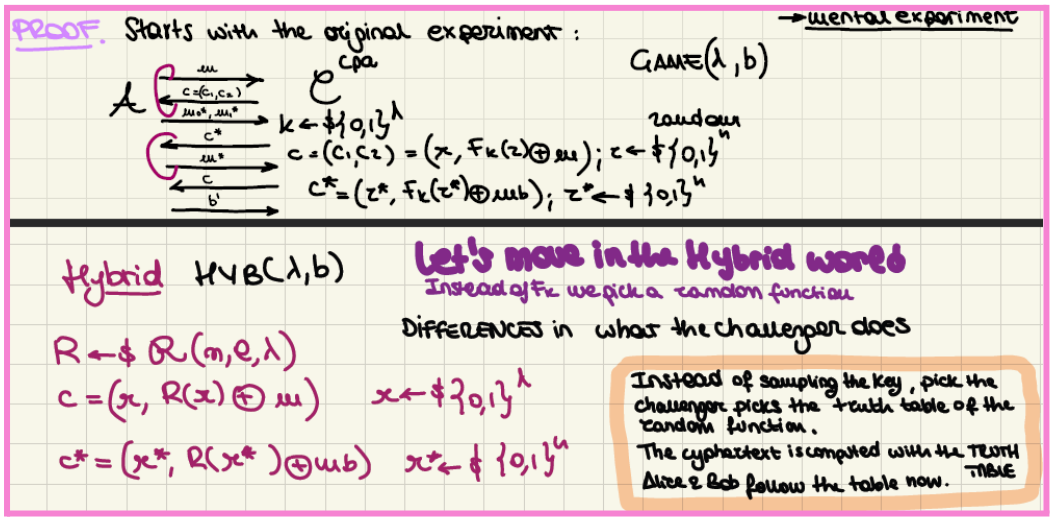
1. PRFs⇒CPA secure SKE
2. OWFs⇒PRFs (PRGs⇒PRFs)

Construction of CPA secure SKE from any PRF family

1. PRFs⇒CPA secure SKE



[Construction] La prima parte del PLAN si basa sul costruire una primitiva Π da $\mathcal{F}$. Il nostro PLAN sarebbe avere una famiglia di funzioni su cui costruire l'encryption e considerare che se questa famiglia di funzioni è pseudorandom allora l'encryption sarà CPA secure.
Il problema è che non possiamo farlo direttamente, perché $F_k(m)$ è deterministica (vedi analisi di Def 15, no deterministic encryption can be CPA secure): dobbiamo quindi prima randomizzarla. Per fare ciò usiamo due componenti c_1 e c_2 come ciphertext e scegliamo l'input (r) random (Recall: the adversary knows r and this is ok, because in Def 16 he can even choose the input). This is the basic idea of CBC, a cryptographic standard cipher block chaining. But what if my message has a different length from l? We will see afterwards. For now FIXED-INPUT LENGTH. Procediamo quindi con il nostro PLAN e definiamo il teorema [Thm 11 - If $\mathcal{F}$ is a PRF family, then Π above is CPA-secure]. Similmente alla dimostrazione di PRG (Thm 7), partiamo dall'esperimento originale $Game(\lambda, b)$ (riscrivendo quindi la definizione di CPA security per questo caso, Def 15) e lo confrontiamo con un esperimento ibrido $Hyb(\lambda, b)$, dove invece di F_k si computa una funzione random R e il ciphertext è computato con la truth table.



Procediamo quindi definendo due lemmi.

- [Lemma 1] $Game(\lambda, b) \approx_c Hyb(\lambda, b),\ \forall b \in \{0,1\}$;

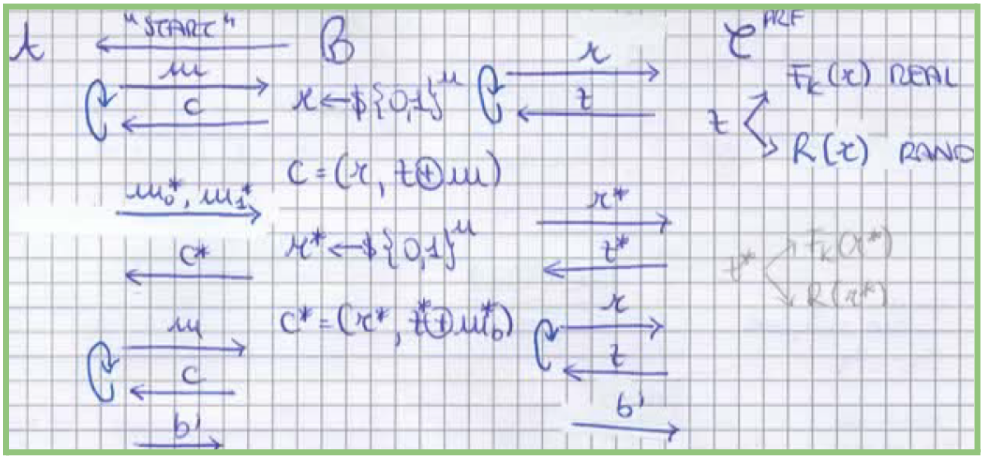
Proviamo il primo lemma, quindi che $Game(\lambda, b)$ e $Hyb(\lambda, b)$ sono computationally close.

(•) We must do a trivial reduction to PRF: fix b, we assume lemma is not true (i.e. the two experiments are distinguishable)

→ then $\exists PPT\ A\ s.t\ |Pr[Game_{\Pi,A}(\lambda, b) = 1] - Pr[Hyb_{\Pi,A}(\lambda, b) = 1]| \geq 1/poly(\lambda)$

→ then $\exists PPT\ B\ s.t\ |Pr[Real_{\mathcal{F},B}(\lambda) = 1] - Pr[Rand_{\mathcal{F},B}(\lambda) = 1]| \geq 1/poly(\lambda)$

(i.e B breaks PRF). So we can proceed to construct B.



Sequence → - B starts A; - A sends m; - B takes random r ($r \to \$\{0,1\}^n$);

- B encrypts m and obtains $c = (c_1, c_2)$ in this way:

- $c_1 = r$
- for c_2 … B sends r to C; C reply with z that can be $z = F_k(r)$ $(Real)$ or $z = R(r)$ $(Rand)$; then $c_2 = m \oplus z$

- B reply to A the ciphertext c for the message m;

(This is repeated $poly$ times)

- A sends m_0^* and m_1^* to B and again is repeated the same process to obtain c^*;
- A asks B for more encryptions again;
- A outputs a bit b'; B sends b' ($b' = b$) to C

By inspection:

In $Real\ z = F_k(r),\ k \leftarrow \$U_\lambda \ \rightarrow Pr[Real_{\mathcal{F},B}(\lambda) = 1] = Pr[Game_{\Pi,A}(\lambda, b) = 1]$

In $Rand\ z = R(r),\ R \leftarrow \$\mathcal{R} \ \rightarrow Pr[Rand_{\mathcal{F},B}(\lambda) = 1] = Pr[Hyb_{\Pi,A}(\lambda, b) = 1]$

By assumption:

$|Pr[Real_{G,B}(\lambda) = 1] - Pr[Rand_{G,B}(\lambda) = 1]| =$

$|Pr[Game_{\Pi,A}(\lambda, b) = 1] - Pr[Hyb_{\Pi,A}(\lambda, b) = 1]| \geq 1/poly(\lambda)$. (contradiction!)

B using A can distinguish the two distributions and so it breaks PRF. This proves the lemma.

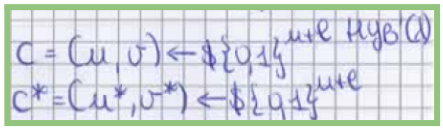
- [Lemma 2] $Hyb(\lambda, b) \approx_s Hyb'(\lambda),\ \forall b \in \{0,1\}$

  [(•) $\approx_s$ stand for statistically close: no machine can distinguish them, not only PPT machines like in computationally indistinguishable]

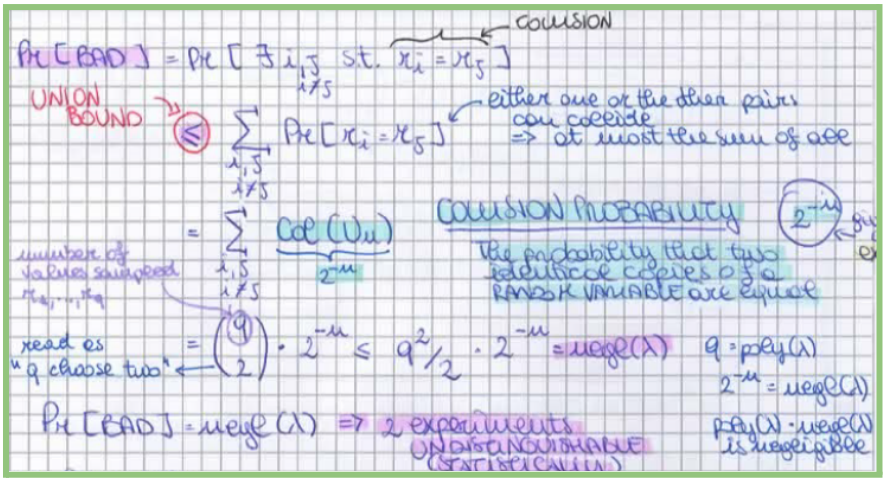
Dobbiamo provare quindi adesso che $Hyb(\lambda, 0)$ è indistinguishable da $Hyb(\lambda, 1)$.

(•) So, now we show that no adversary can win in Hyb because this is like OTP. There is still something that I can do wrong: pick twice the same r; but in this way it is no more like OTP. This event will not happen, but we need to prove it.

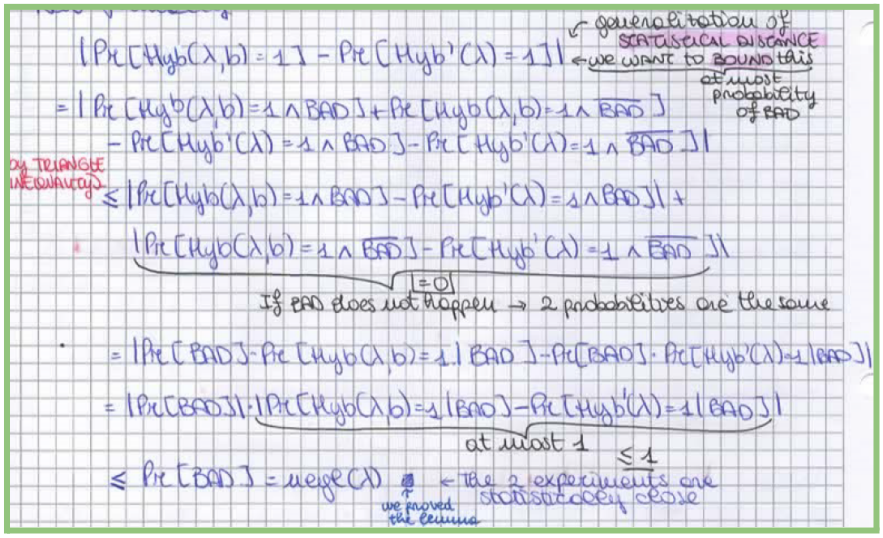
Chiamiamo *BAD* event l'evento non fortunato dove viene presa la stessa r per due ciphertext differenti. Introduciamo un game $Hyb'(\lambda)$ dove c_1 e c_2 sono entrambi random (diversamente da $Hyb(\lambda, b)$ dove $c = (c_1, c_2) = (r, R(r) \oplus m), r \leftarrow \$\{0,1\}^n$) e dimostriamo con il secondo lemma che è statistically close a $Hyb(\lambda, b)$. In generale, se due esperimenti sono uguali eccetto per un qualche *BAD* event, per provare che sono indistinguibili basta provare che la probabilità del *BAD* event è negligible. So let *BAD* be the event that the sequence of r are not distinct ($r_1, r_2, ..., r_q$ are all the values ever sampled, with $q = poly(\lambda)$).



(•) If r are all distinct, we run R on a sequence of distinct input which means we only explore distinct rows of the table. Then for every row, the output is random and so the distribution of c_2 is random (because random⊕constant=random). So unless something bad happens ($\neg BAD$), the distribution of c_2 is uniform ($Hyb' \equiv Hyb$). So all that remains to do is to bound the probability of *BAD*.



Now formally:



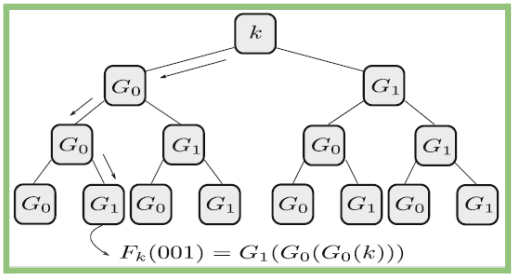
Possiamo quindi infine concludere la dimostrazione del teorema grazie ai due lemmi: $Game(\lambda, 0) \approx_c Hyb(\lambda, 0) \approx_s Hyb'(\lambda) \approx_s Hyb(\lambda, 1) \approx_c Game(\lambda, 1)$.

- (Construction +) Thm 11 - If $\mathcal{F}$ is a PRF family, then Π above is CPA-secure (+ proof) + Lemma 1 (+ proof) + Lemma 2 (+ proof)

(CRYPTO 2022)
Nel 2022, dimostrazione completa del Lemma 1 (nel 2021, Exercise).
Nel 2022, esplicitato come lemma il fatto che Hyb' non dipende dal bit.
[Lemma] $Hyb'(\lambda, 0) \equiv Hyb'(\lambda, 1)$
Proof. Because Hyb' doesn't depend on b (the challenger sends random string so the distribution doesn't depend on b).

Constructing PRFs from PRGs: The GGM construction

2. OWFs⇒PRFs (infatti PRGs⇒PRFs)
[Construction] La seconda parte del PLAN si basa sulla costruzione GGM. Sia $G: \{0,1\}^{\lambda} \to \{0,1\}^{2\lambda}$ (a PRG that doubles the length of its arguments) e $G(s) = (G_0(s), G_1(s))$ (dove $G_0(s)$ sono i primi λ bit di $G(s)$ e $G_1(s)$ i secondi λ bit), si osserva che $F_k(x) = G_x(k)$ è una PRF per input uguale a 1 (for domain $\{0,1\}$). Questo è inoltre dimostrabile costruendo la truth table e notando che adesso è una random truth table, quindi una PRF. Ora bisogna estendere la casistica ad un input maggiore di un bit. Possiamo pensare alla tabella come ad un albero, il GGM Tree, dove se $x = 0$ vado a sinistra e se $x = 1$ a destra. In generale $F_k(x) = G_{x_n}(G_{x_{n-1}} \dots G_{x_2}(G_{x_1}(k)) \dots)$ per ogni $n = poly(\lambda)$. Si arriva così a definire mediante il GGM Theorem [Thm 12 - If G is a PRG, F is a PRF] l'obiettivo di questa seconda parte del PLAN.



- (Construction +) Thm 12 - If G is a PRG, F is a PRF

Lecture 8

[1]: 5.2, 7.1, 7.2 [2]: 4.1, 4.2, 4.3, 7.5 [7]: 3.2

- ➢ Proof of security for the GGM construction.
- ➢ Message authentication codes in the computational setting: Unforgeability against chosen-message attacks.
- ➢ Proof that every PRF yields a fixed-input length MAC.

(CRYPTO 2022)
Nel 2022, non dimostrazione formale di OWFs ⇒ PRFs (PRGs ⇒ PRFs) (Thm 12)
We need to prove that F_k behaves like a truly random f, indistinguishable from a truly random f, so for every x it outputs λ bits at random, but it's deterministic. The idea is to prove (like with the hybrid argument) that a GGM tree is indistinguishable from a tree where everything is random. Unfortunately, we cannot do this because the node number is exponential and this implies that we can only explore a polynomial number of paths without putting randomness inside each layer. It works for $n = 1$ (I can construct a GGM tree with k as root and G_0 and G_1 as leaves and see that the construction is computational indistinguishable from a random truth table). Then we assume that it is true for n and we prove for $n + 1$ (we don't do this, but it works).

(Proof Thm 12) In particolare, la dimostrazione si basa su un lemma [Lemma] che prova essere uniforme per ogni seed una PRG che prende t seeds indipendenti. Ora per provare il teorema, ci basiamo su un altro lemma [Lemma] (dimostrazione basata su hybrid experiments da $Real$ a $Rand$ e su due claim) che prova (inductive step) che presa una PRF F_k' e una PRG G, allora F_k funzione che simula uno step del GGM è una PRF. Una volta fatti i due claim sulla relazione tra gli hybrid game abbiamo provato il lemma e così anche il teorema iniziale.

- … + Lemma 1 + Lemma 2 (+ proof + CLAIM 1 with proof as Exercise + CLAIM 2 with proof)

RECAP
Shannon theory → It's possible to do perfectly secure encryption although with some limitations. Our GOAL was to overcome these limitations → we're almost there!
Assuming any OWF, we can construct encryption using a short key, independently from the message length, that can be used to encrypt many arbitrary messages.
Open questions (we will answer in the future):

- How to encrypt long messages? PRF→ can encrypt messages with bounded length (same length of PRF output).
  IDEA: extend input/output of PRF→ but how do we preserve security?
- How does encryption look in real life?
- Can we get something stronger than CPA security? Yes→ real world protocols were attacked with something that was stronger than CPA security.

Let's take a break from encryption and let's do message auth efficiency!

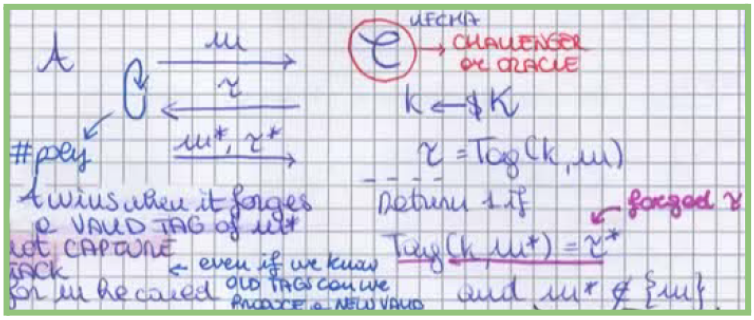
Message Authentication Codes (2)

Possiamo fare secrecy e authenticity allo stesso tempo?

RECALL: Security notion for Tag ($Tag: KxM \to \tau$) → One Time Security (information theoretical): problem, cannot get a short key, the key length depends on the number of messages that you want to authenticate.
We want to go beyond what we have seen → power of computational assumptions.

Computationally secure MACs

Definiamo gli Unforgetability - Chosen Message Attacks [Def 17 - UF-CMA] per andare oltre a quello che abbiamo fatto in precedenza, sfruttando la computational assumption. A $\Pi = Tag$ is UF-CMA if $\forall PPT\ A,\ Pr[Game_{\Pi,A}^{UFCMA}(\lambda) = 1] \leq negl(\lambda)$. Nel game della definizione, l'avversario invia prima della sfida (poly times) messaggi m al challenger, che risponde con tag validi τ associati a ciascun messaggio. L'avversario vince se riesce a forgiare poi un tag valido τ^* per un messaggio m^*. In sostanza l'avversario per vincere deve riuscire a capire la funzione di tag. Non si può realizzare un reply attack (if A knows a tag for m, he could reply with this), perchè l'avversario manda sempre un messaggio nuovo (freshness for the message, $m^* \notin \{m\}$).

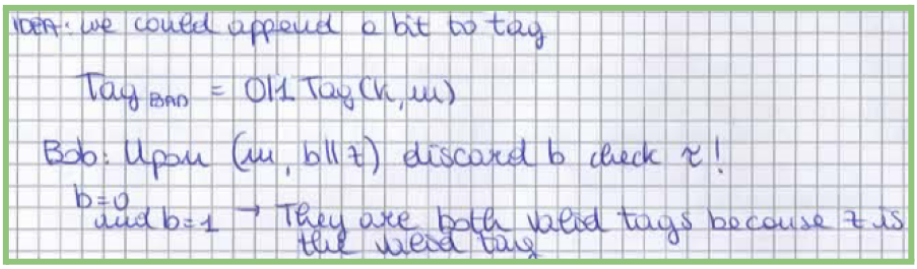


- Def 17 - UF-CMA

[telegram]: 9.1
The desirable property that a mac scheme should hold is to prevent any attacker from generating a valid couple (m, τ), even after querying a tagging oracle polynomially many times. The act of generating a valid couple from scratch is called forging, and the aforementioned property is defined as unforge-ability against chosen-message attacks (or uf-cma, in short). Do note that m is stated to be outside the query set M, expressing the "freshness" of the forged couple.

Exercise (similar to exam, + idea). Consider strong UF-CMA: A is allowed to forge on $m^* \in \{m\}$ so long as $\tau^* \neq \tau$. (1) Formalize this notion (write a formal definition). (2) Prove (by reduction) or disprove (you give an example/by separation) that UF-CMA⇔STRONG UF-CMA.
(•) Suggestions:
- STRONG UF-CMA⇒UF-CMA, but UF-CMA⇏STRONG UF-CMA;
- For separation you need to find a tag that is STRONG UF-CMA, but not UF-CMA. (Nota. By separation → you give an example for which some property doesn't hold. Make it simple by "cooking it at", like constructing your case (giocare con lunghezza dei messaggi o simili) instead of proving the most famous case (you are not in the real world).

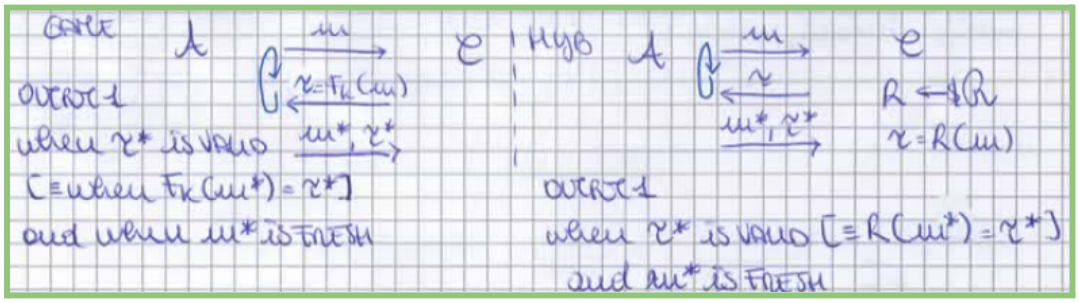
UF-CMA ⇏ STRONG UF-CMA. To show that *A*⇏*B*, we must find something in *B* that does not work for *A* (*BAD*) so $\exists Tag_{BAD}$ such that Tag_{BAD} is UF-CMA but not STRONG UF-CMA. Main idea: start with UF-CMA Tag and use it to design Tag_{BAD}. In order to break STRONG UF-CMA there must exists m with associated two tags τ and τ' , both valid.



So we must prove Tag_{BAD} is UF-CMA (reduction to Tag) but not STRONG UF-CMA (attack).

Every PRF yields a FIL MAC

Definiamo ora il teorema per cui data una PRF family $\mathcal{F} = \{F_k\{0.1\}^n \to \{0.1\}^l\}_{k\epsilon\{0.1\}^\lambda}$, allora $Tag(k,m) = F_k(m)$ è UF-CMA, fintanto che (so long as) $l(\lambda) = \boldsymbol{\omega}(log(\lambda))$ ($\boldsymbol{\omega}(log(\ldots) \to$ SUPER LOGARITHMIC) [Thm 13 - PRF⇒MAC, let $\mathcal{F} = \{F_k\{0.1\}^n \to \{0.1\}^l\}_{k\epsilon\{0.1\}^\lambda}$ be a PRF family, then $Tag(k,m) = F_k(m)$ is UF-CMA, so long as $l(\lambda) = \boldsymbol{\omega}(log(\lambda))$]. Se supponessimo che $l(\lambda)$ fosse un logaritmo e non un super logaritmo il teorema sarebbe falso perché non sarebbe negligible la probabilità; invece, con il superlogaritmo è negligible. La dimostrazione del teorema avviene definendo l'esperimento $Game$, in cui τ viene computata dal challenger con una PRF, e l'esperimento Hyb, in cui τ viene computata tramite una funzione random R.



La dimostrazione può proseguire poi in due maniere differenti:

- (A) Reduction with distinguisher, with one lemma

[Lemma 1A] $Game \approx_c Hyb$. You can prove it with a distinguisher: assume there is a distinguisher and turn it into an adversary that breaks this game.
Exercise. Prove the lemma by reduction.

- (B) Reduction with forger, with two lemmas

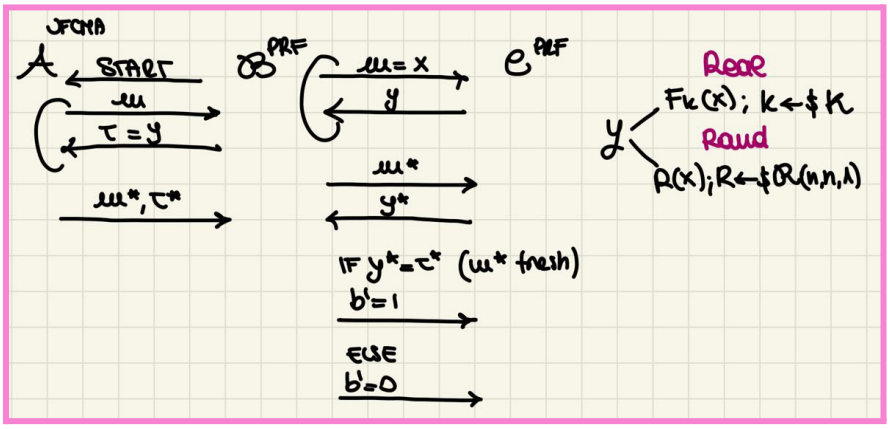
[Lemma 1B] $\forall PPT\ A$: $|Pr[Game(\lambda) = 1] - Pr[Hyb(\lambda) = 1]| \leq negl(\lambda)$ → This lemma says that the probability that A wins in $Game$ and in Hyb are negligible close. Here the adversary is a forger (and not a distinguisher).
Exercise. Prove the lemma by reduction
[Lemma 2B] $Pr[Hyb(\lambda) = 1] \leq negl(\lambda)$, $\forall$ unbounded $A \to A$ in order to forge should know the output of a random function $R(m^*)$ at a point that he cannot query; but query does not help me. A can only guess because it's output of a random variable. Note: $\forall A$, $Pr[Hyb(\lambda) = 1] \leq 2^{-l}$; $2^{-l} = negl(\lambda)$ when $l(\lambda) = \boldsymbol{\omega}(log(\lambda))$.
Possiamo affermare infine quindi che PRF ci da direttamente un MAC e quindi MAC esiste data ogni PRF.

- Thm 13 - PRF⇒MAC, let $\mathcal{F} = \{F_k\{0.1\}^n \to \{0.1\}^l\}_{k\in\{0.1\}^\lambda}$ be a PRF family, then $Tag(k,m) = F_k(m)$ is UF-CMA, so long as $l(\lambda) = \omega(log(\lambda))$ (+ proof) + Lemma 1A (+ proof as Exercise) or Lemma 1B (+ proof as Exercise) + Lemma 2B (+ proof)

(CRYPTO 2022)
Nel 2022, enunciato Thm 13 senza il superlogaritmo; segue dimostrazione (simile ad approccio B del 2021).

[Thm C - PRF⇒MAC, let $\mathcal{F} = \{F_k\{0.1\}^n \to \{0.1\}^l\}_{k\in\{0.1\}^\lambda}$ be a PRF family, then $Tag(k,m) = F_k(m)$ is UF-CMA] This means that if you have to tag a message of fixed length, you can use AES. PRF is our abstraction of AES (remember AES is called cipher but it's not a cipher, but a PRF).
Proof. We prove the theorem by reduction: we assume…

- $\exists PPT\ A$ such that wins the UF-CMA game with probability at least $1/poly(\lambda)$;
- then construct B that breaks PRF.



Intuition: when we are in $Real$, $y = F_k(x)$ so the simulation is perfect and so B will win with probability at least $1/poly(\lambda)$.
Analysis: until now, we proved that $Pr[Real_B(\lambda) = 1] = Pr[Game_A(\lambda) = 1] \geq 1/poly(\lambda)$, but this is not sufficient because we didn't prove that B breaks PRF.
Now we need to compute $Pr[Rand_B(\lambda) = 1]$:

$Pr[Rand_B(\lambda) = 1] = Pr[y^* = \tau^* : y^* = R(m^*)] \leq 2^{-n(\lambda)} = negl(\lambda)$.

Then, $|Pr[Real_B(\lambda) = 1] - Pr[Rand_B(\lambda) = 1]| \geq 1/poly(\lambda) - negl(\lambda) = 1/poly(\lambda)$.

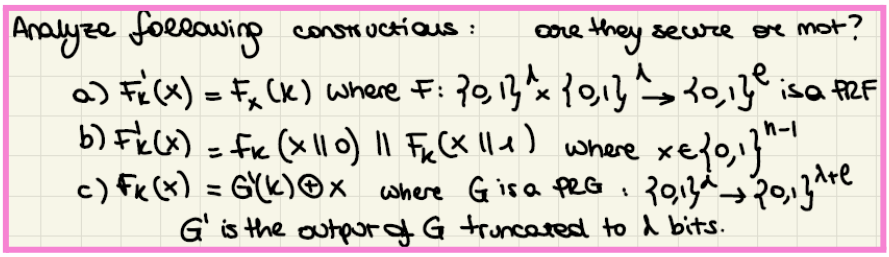
(Contradiction!)

- Thm C - PRF⇒MAC, let $\mathcal{F} = \{F_k\{0.1\}^n \to \{0.1\}^l\}_{k\in\{0.1\}^\lambda}$ be a PRF family, then $Tag(k,m) = F_k(m)$ is UF-CMA (+ proof)

So we have proved that every PRF yields a fixed-input length MAC. So MAC exists given any PRF. How do we tag LONG MESSAGES? We will show that it is possible to arbitrarily extend the domain of a PRF (CBC MAC construction allows us to do this).

Exercise (+ idea). Prove that no PRF can be secure against computationally unbounded adversaries. Strategy 1: given PRF → PRG and proof follows from some exercise for PRG. Strategy 2: directly prove this.

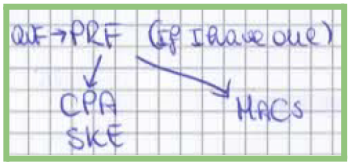
Exercise.



Lecture 9

[1]: 7.3 [2]: 4.3, 4.4 [7]: 3.3

- ➢ Domain extension for PRFs via almost-universal hash functions.
- ➢ Constructions of almost universal hash functions.
- ➢ CBC-MAC and Encrypted CBC-MAC, XOR-MAC.



What we have done until now? Siamo partiti dalla OWF. Abbiamo dimostrato poi che OWF⇒PRF (OWF⇒PRG⇒PRF). Dalla PRF abbiamo dimostrato che possiamo passare sia su CPA SKE (PRF⇒CPA SKE, Lecture 7) che su MAC (PRF⇒MAC, Lecture 8). In entrambi i casi possiamo superare Shannon (one short key to encrypt many messages / one short key to authenticate many messages).
Limitation: fixed input length (limite che avevamo già dato a PRF).
What do we have to do now?

- Variable input length per CPA SKE e MAC.
- Come funziona PRF nel real world?
- How to get CPA SKE+MACs? → Authenticated encryption (SECURE CHANNEL = AUTHENTICATE + ENCRYPT).

In questa lezione ci occupiamo di MAC con variable input length.

Domain extension for PRFs via almost-universal hash functions

Domain extension → Dobbiamo estendere la lunghezza del nostro input (per MAC oggi e per CPA SKE nel futuro). Sia m un lungo messaggio sequenza di messaggi più corti ($m = (m_1, m_2, \ldots, m_d)$). Assumiamo che la lunghezza di questo sia un multiplo di n ($m_i \in \{0.1\}^n$). Come autentico m?

Failed attempts:

- XOR all blocks: $\tau = Tag(k, \oplus_{i=1}^{d} m_i)$

  Nel primo metodo facciamo lo XOR di tutti i blocchi. Problema: Tag non è sicura, anche per fixed input length, e anche se la funzione Tag è sicura. Perchè? Si prova che creando un m' con i messaggi di m ordinati differentemente, si ottiene lo stesso tag.
- $\tau_i = Tag(k, m_i); \tau = (\tau_1, \ldots, \tau_d)$

Nel secondo metodo applichiamo Tag a ciascun messaggio, ottenendo una sequenza di tag. PROBLEMA. Potrei fare lo swap di due messaggi con i relativi Tag e ottenere ancora un messaggio autenticato, trovando così Tag.

- $\tau_i = Tag(k, m_i||i)$

  Nel terzo metodo autentichiamo anche la posizione: aggiungiamo il block number al messaggio (more bits, n bit del messaggio + bit per rappresentare la sua posizione), "uccidendo" così lo swapping. PROBLEMA. Comunque si può rompere perché posso prendere parti di un messaggio e parti di un altro.

[2]: 4.4 (Extension to Variable-Length Messages)

1. *XOR all the blocks together and authenticate the result.* I.e., compute the tag $t := \mathsf{Mac}'_k(\bigoplus_i m_i)$. In this case, an adversary can forge a valid tag on a new message by changing the original message so that the XOR of the blocks does not change. This can easily be done.

2. *Authenticate each block separately.* I.e., compute $t_i := \mathsf{Mac}'_k(m_i)$ and output $\langle t_1, \ldots, t_d \rangle$ as the tag. This prevents an adversary from sending any previously-unauthenticated block without being detected. However, it does not prevent an adversary from changing the order of the blocks, and computing a valid tag on, e.g., the message $m_d, \ldots, m_1$ (something that is not allowed by Definition 4.2).

3. *Authenticate each block along with a sequence number.* I.e., compute $t_i := \mathsf{Mac}'_k(i \| m_i)$ and output $\langle t_1, \ldots, t_d \rangle$ as the tag. This prevents the re-ordering attack described above. However, the adversary is not prevented from dropping blocks from the end of the message (since $\langle t_1, \ldots, t_{d-1} \rangle$ is a valid tag on the message $m_1, \ldots, m_{d-1}$). Furthermore, the adversary can mix-and-match blocks from different messages. That is, if the adversary obtains the tags $\langle t_1, \ldots, t_d \rangle$ and $\langle t'_1, \ldots, t'_d \rangle$ on the messages $m = m_1, \ldots, m_d$ and $m' = m'_1, \ldots, m'_d$, respectively, it can output the valid tag $\langle t_1, t'_2, t_3, t'_4, \ldots \rangle$ on the message $m_1, m'_2, m_3, m'_4, \ldots$

Our solution. Arriviamo alla nostra soluzione definendo una input shrinking function $h_s: \{0,1\}^N \to \{0,1\}^n$ (hash function family, $\mathcal{H} = \{h_s: \{0,1\}^N \to \{0,1\}^n\}_{s \in \{0,1\}^\lambda}$), con $N >> n$ (very long input $n * d$). Questa funzione lavora prendendo un input molto grande e restituendo qualcosa di piccolo (*to shrink*, restringere). Così, anche se la lunghezza dell'input è tanto grande, possiamo avere una lunghezza fissa più piccola da cui creare il tag ($Tag_k(h_s(m))$), risolvendo così il problema di variabilità. Nota: lo XOR in precedenza era una input shrinking function, ma non era buona.

Quale security abbiamo da h_s? Esistono collisioni?

Poiché $N >> n$, differenti input potrebbero essere mappati su uno stesso output: *if I can find* $m, m' \, with \, m \neq m' \, s.t. \, h_s(m) = h_s(m')$, *we are dead!*

Abbiamo quindi due opzioni ([telegram]: 9.2.1).

- Assumere che sia molto difficile trovare collisioni, anche se il seed $s \in \{0,1\}^\lambda$ fosse pubblico; in tal caso esiste una famiglia di CRH - Collision Resistant Function (Lecture 12) che assicura che non ci siano collisioni (perchè $\mathcal{H}$ molto grande).
  (•) Known fact: OWF⇏CRH. So we can't construct collision resistant hash functions from OWF: we go out of minicrypt and we will do later with number theory.
- Seed segreto: se s è segreto è difficile capire come h_s funziona (it's simple information-theoretically).

(•) Alice and Bob go to the secret place and they exchange the key. For this construction we have one key for the PRF and s that is the seed of the hash functions.

Per adesso rimaniamo nel mondo della Minicrypt e in questa lezione ci focalizzeremo quindi sulla seconda opzione (seed s segreto).

(•) RECAP. We were studying the problem of extending the domain of PRF. In particular we have shown that every PRF gives a FIL MAC by itself. Now the question is what if I have a very long message that we wish to MAC? We saw some natural approaches that didn't work and we ended up with this approach: 1. we start with a very long message; 2. we hash it; 3. we input the output of the hash function to the PRF.
Here there are two possibilities:
- assume that the hash function is secret; - assume that the hash function is public.
Now we will do the first one because public hash functions require stronger assumptions than what we have currently studied. So we are gonna use secret hash function and it's fine to use it in this case because we are doing secret key cryptography, and so Alice and Bob can share the key for the hash function. Moreover, secret key hash functions can be realized without any computational assumptions.

(CRYPTO 2022)
Nel 2022, Lecture 11 fatta da remoto su zoom, seguendo documento [7], in particolare [7]: 3.3 (Domain Extension).

- Definition 16 (Universal Hash Function) corrisponde a Def 18.

Definition 16 (Universal Hash Function). The family of functions $\mathcal{H} = \{h_s : \{0,1\}^N \to \{0,1\}^n\}_{s\in\{0,1\}^\lambda}$ is universal (as in Universal Hash Function (UHF)) if for all distinct x, x' we have that

$$\Pr_{s\leftarrow\$\{0,1\}^\lambda}[h_s(x) = h_s(x')] \leq \varepsilon.$$

Two cases are possible, depending on what ε is:

- if $\varepsilon = 2^{-n}$, then $\mathcal{H}$ is said to be Perfect Universal (PU);
- if $\varepsilon = \text{negl}(\lambda)$, with $\lambda = |s|$, then $\mathcal{H}$ is said to be Almost Universal (AU). ◇

With UHF we can extend the domain of a PRF.

Now with the concept of universal hash function we can extend the domain of PRF.

- Theorem 14 corrisponde a Thm 14.
- Construction 7 (UHF with Galois field) corrisponde alla nostra costruzione di AU hash function tramite information theoretically

GF: Nota. Questa construction funziona anche se field non di Galois.

Construction 7 (UHF with Galois field). Let $\mathbb{F}$ be a finite field, such as the Galois field over 2^n. In the Galois field, a bit string represents the coefficients of a polynomial of degree $n-1$. Addition is the usual, while for multiplication an irreducible polynomial $p(x)$ of degree n is fixed, and the operation is carried out modulo $p(x)$.

We pick $s \in \mathbb{F}$, and $x = x_1||\dots||x_t$ with $x_i \in \mathbb{F}$ for all i. The hash function is defined as

evaluation of the hash function — variable of the polynomial — coefficient

$$h_s(x) = h_s(x_1||\dots||h_t) = \sum_{i=1}^{t} x_i \cdot s^{i-1} = Q_x(s).$$

HOW TO PROVE IT'S UNIVERSAL?

A collision is two distinct x, x' such that

s needs to be a root of the polynomial

$$Q_x(s) = Q_{x'}(s) \iff Q_{x-x'}(s) = 0 \iff \sum_{i=1}^{t}(x_i - x'_i)s^{i-1} = 0.$$

This means that s is a root of $Q_{x-x'}$. So the probability of a collision is:

root numbers — # possible cases / # total cases

$$\Pr[h_s(x) = h_s(x')] = \frac{t-1}{|\mathbb{F}|} = \frac{t-1}{2^n} \cdot \frac{\#\text{ POSSIBLE CASES}}{\#\text{ TOTAL CASES}} \quad \text{(negligible)}$$

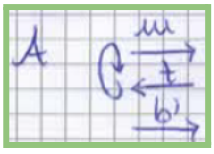
- Costruzione di *computational variants of hash function.*
- CBC-MAC. Theorem 15 (CBC-MAC is a computationally secure AU hash function if F is a PRF), Theorem 16 (CBC-MAC is a PRF) e Theorem 17 (CBC-MAC is AU) corrispondono a Lemma - CBC-MAC defines completely an AU family e Thm 15
- XOR-MAC. Thm 16 non enunciato come teorema nel 2022

Definiamo quindi ora una famiglia $\mathcal{H}$ con probabilità negligible di collisione sulla scelta di s [Def 18 - Almost Universality]: almost universality, $\mathcal{H}$ is ε-AU if $\forall m, m' \epsilon \{0,1\}^N$, $m \neq m'$, $Pr[h_s(m) = h_s(m')] \leq \varepsilon = negl(|s|)$, $s \to \${0,1}^{\lambda}$). È una sorta di collision resistant (*information-theoretically*, or statistical). Si definisce perfect universality se ε è 2^{-n}.

- Def 18 - Almost Universality

Questo approccio può essere adottato per fare domain extension (VIL) di MAC ($Tag_k(h_s(m))$) ma anche di PRF. Questo ci è dato dal teorema [Thm 14 - $\mathcal{F}(\mathcal{H}) = \{F_k(h_s(\cdot))\}$ is a PRF from $N \to l$, assuming $\mathcal{H}$ is AU and $\mathcal{F}$ is a PRF from $n \to l$].

Questo teorema ci permette quindi di estendere il dominio di PRF e, dato che PRF⇒MAC (Thm 13), di estendere automaticamente anche il dominio di MAC. Per la dimostrazione (in figura, recall dello schema del game) consideriamo tre esperimenti:



- $Real \to z$ computata con la nostra funzione h_s, $z = F_k(h_s(m))$,
- $Rand \to z$ computata con R ottenuta da truth table, $z = R(m)$,
- $Hyb \to z$ computata con la nostra h_s in input ad R, $z = R(h_s(m))$.

Si prosegue con due lemmi.

- [Lemma 1] $Real \approx_c Hyb$

(•) It can be proved by reduction. Because the PRF is secure, you cannot distinguish the $Real$ from Hyb (all we did in Hyb was to replace the PRF with the random function).
Exercise. Prove the lemma by reduction to F_k.

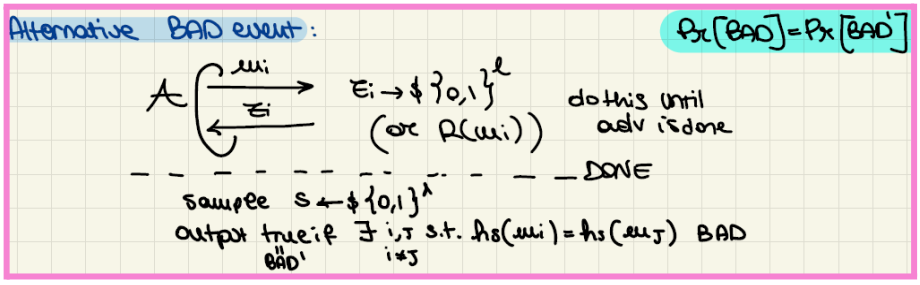
- [Lemma 2] $Hyb \approx_s Rand$.

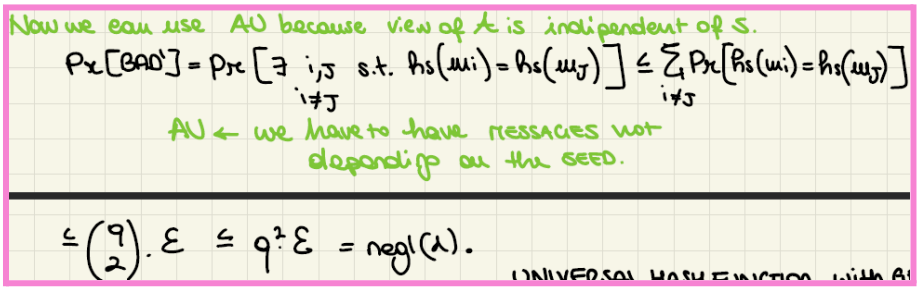
Per la dimostrazione di quest'ultimo, consideriamo il *BAD* event che una collisione possa accadere in Hyb (collision → we can distinguish).

(•) Intuitively, if the adversary never finds a collision for the hash function in the Hyb, the function R is always evaluated on fresh inputs (inputs that are always different). So the output is always gonna be random and independent for each input which is the same as in $Rand$. So the bad thing that can happen is that the adversary finds a collision. If this event doesn't happen, then the two experiments are identical, which means that the probability of distinguishing the two experiments is at most the probability of the *BAD* event.

Come nella prova del secondo lemma di PRF⇒CPA-secure (Thm 11), dobbiamo dimostrare che la probabilità di BAD sia negligible. Proviamo ciò cambiando l'esperimento, in modo tale da sfruttare la definizione di Almost Universality (Def 18) (nota: per poter utilizzare AU, è necessario avere messaggi non dipendenti dal seed).

(•) In order to use almost universality, you need that the inputs to the hash functions are independent of s because in the definition of universal hash function you first fix m and m' and then you sample s. Here instead you cannot immediately say that m is independent of s, because in the experiment the challenger first sample s and then uses it in order to reply to these queries $m_1, m_2, \ldots, m_q$. However we can easily redefine the experiment in a way that allows us to use universality. Intuition: in order to compute the probability of *BAD*, you can think of a different experiment in which you do not sample s at the beginning, so the adversary makes the queries and then the challenger replies with completely independent random values (essentially with R). Now, after the adversary has done all the queries, you sample the seed s and you check if the event happens. You can do it afterward and it doesn't change the probability of *BAD*, because until the *BAD* doesn't happen, the adversary only sees random values. If you sample s at the end, the input are independent of s and so you can use the definition of universality.





- Thm 14 - $\mathcal{F}(\mathcal{H}) = \{F_k(h_s(\cdot))\}$ is a PRF from $N \to l$, assuming $\mathcal{H}$ is AU and $\mathcal{F}$ is a PRF from $n \to l$ (+ proof) + Lemma 1 (+ proof as Exercise) + Lemma 2 (+ proof)

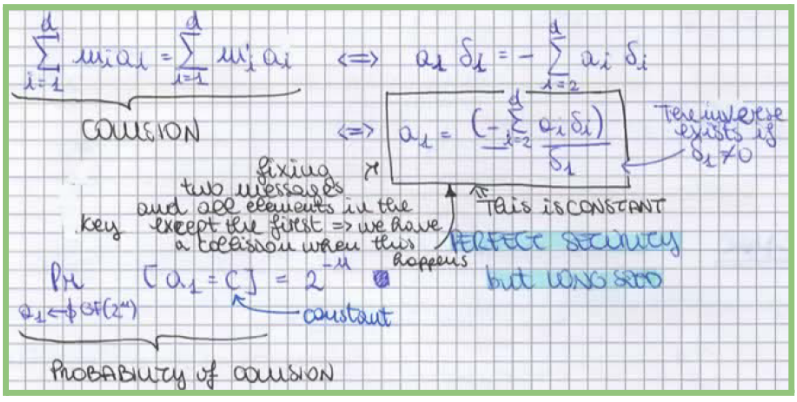
Constructions of almost universal hash functions

Procediamo con la costruzione di AU hash families (Recall: AU → we have to have messages not depending on the seed).

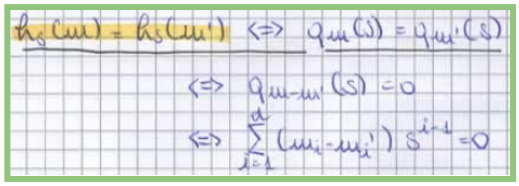
Ci sono due costruzioni: 1. Information theoretically; 2. Computationally .

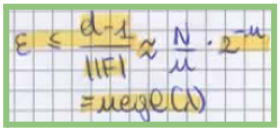
1. Information theoretically

Nota: Galois Field (GF), o finite field, è un campo (field) che contiene un numero finito di elementi (un field è un set in cui addizione, sottrazione, moltiplicazione e divisione sono definite e soddisfano il field axioms). Prendiamo $\mathbb{F} = GF(2^n)$ un campo finito di 2^n elementi, $m = (m_1, m_2, \dots, m_d) \in GF(2^n)$ e $s = (a_1, a_2, \dots, a_d) \in \mathbb{F}^d$ (seed grows with the number of blocks → LONG SEED). Definiamo la nostra hash function $h_{a_1,a_2,\dots,a_d}(m_1, m_2, \dots, m_d) = \Sigma_{i=1}^{d} a_i m_i$ (inner product). Prendendo due messaggi, si può dimostrare che la probabilità di collisione su questi sia costante, uguale a 2^{-n}: abbiamo quindi PERFECT SECURITY, ma con un LONG SEED.



Let's do better! Prendiamo $\mathbb{F}$ e m come prima. Il seed invece lo computiamo questa volta randomicamente ($s \leftarrow \$\mathbb{F}$). Costruiamo l'hash family function: $h_s(m) = \Sigma_{i=1}^{d} s^{i-1} m_i = q_m(s)$ ($q_m(s)$ è un polinomio di grado $d-1$). Dobbiamo dimostrare quindi che h_s è AU. Abbiamo collisioni se e solo se s è root (radice, zero) del polinomio (punto in cui la funzione si annulla). Se il grado è $d-1$, si avranno al più $d-1$ roots. Dato che il numero di punti è $|\mathbb{F}| = 2^n$, allora proviamo che ε è negligible. Abbiamo quindi qui una SHORT SEED, ma NON una PERFECT SECURITY (solo almost universality).





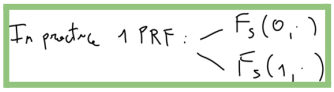
È possibile avere PERFECT SECURITY + SHORT SEED, ma non lo mostreremo.

2. Computationally

We now look at a computational variant of hash functions. We want hash functions for which collisions are difficult to find for any PPT adversary A (computationally bounded).

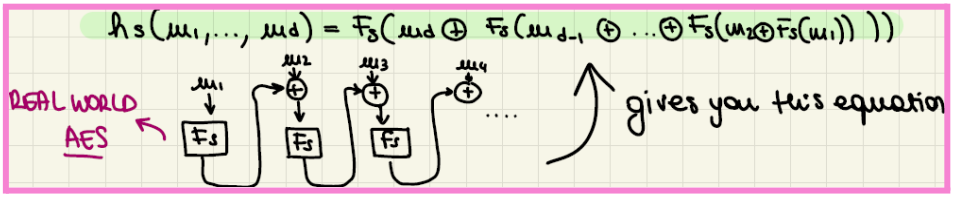
Un altro modo per ottenere domain extension per MAC consiste nel costruire una hash function family usando una famiglia di PRF $\mathcal{F} = \{F_s\{0.1\}^n \rightarrow \{0.1\}^n\}$. Ne usiamo due: una per fare l'estensione del dominio ed una per costruire $\mathcal{H}$.

Separate the domain → You have just one PRF family and then if the input starts with 0 you think of it as the first PRF and if the input starts with 0 as the second PRF, so you sacrifice the first bit of the input.



CBC-MAC and Encrypted CBC-MAC

Il Cipher Block Chain (CBC) è un mode of operation, nato per fare encryption ma che può essere usato anche per MAC (CBC construction and mode of operations, Lecture 11). Lo useremo per costruire una computationally AU family. Construction: you divide your input m in blocks $m_1, m_2, \dots, m_d$. Now you input m_1 to F_s and the output is xored with m_2. Then you input this to F_s and you do the xor with the next block and so on …



Definiamo un lemma [Lemma - CBC-MAC defines completely an AU family] per cui $\mathcal{H}$ definita come sopra è computationally AU (il lemma può essere provato con le proprietà di PRF, we won't see it). Quindi questo significa anche (by the theory studied) che $\mathcal{F}(\mathcal{H})$ così definita è un MAC e definiamo questa costruzione Encrypted CBC-MAC (E-CBC-MAC). E-CBC-MAC funziona anche per variable input length (VIL). Nel mondo reale la PRF F_s è AES. Il teorema [Thm 15 - CBC-MAC directly gives a FIL MAC] ci dice che che $\mathcal{H}$ è direttamente un MAC, ma solo per fixed input length.

(•) CBC-MAC può essere sicuro solo per FIL, ma …

(*a*) se $r \neq 0^n$, $\tau = (r, c_t)$ oppure (*b*) se $r = 0^n$ ma $c_1, \dots, c_t$ è l'output (invece di solo c_t) allora è insicuro (we mean UF-CMA security) anche per FIL.

Exercise. Show CBC-MAC is insecure if (a) or (b).

Exercise (+ solution). CBC is not VIL secure (i.e. exists an adversary that can make a forgery with messages of different length).

(•) The adversary can ask to tag a message m_1 (it is a simple block), obtaining $F_k(m_1)$. Now the adversary can forge a message $m_2 = m_1 || (\varphi_1 \oplus m_1)$ (a message of two blocks): indeed $CBCMAC(m_2) = F_k((m_1 \oplus \varphi_1) \oplus F_k(m_1)) = F_k(m_1 \oplus \varphi_1 \oplus \varphi_1) = F_k(m_1) = \varphi_1$. So given (m_1, φ_1), φ_1 is a valid tag for m_2 which means that CBC-MAC is not VIL secure MAC.

- Lemma - CBC-MAC defines completely an AU family
- Thm 15 - CBC-MAC directly gives a FIXED INPUT LENGTH MAC

(•) Summary. We have studied that if you want to have a MAC for larger messages what you can do is to extend the domain of a PRF, because PRF implies MAC (Thm 13). In particular, you can start with a short input PRF and get a larger input PRF by using a computationally almost universal hash family, where the hash function is secret. This is what the theory tells us. In practice, we use CBC-MAC and this exactly follows this paradigm, because we can prove that this construction is a computationally secure almost universal hash function. Now we can construct $\mathcal{F}(\mathcal{H})$ and we get a long input PRF and thus a long input MAC.

Actually, we can prove that if we do that, we get not only a FIL MAC but also a VIL MAC. The CBC-MAC, by itself, is either a computational secure almost universal hash function (Lemma - CBC-MAC defines completely an AU family) or it is a FIL PRF (so FIL MAC, Thm 15). If you add another layer of encryption, meaning that you do $\mathcal{F}(\mathcal{H})$ in the case of CBC-MAC, you

get the so-called Encrypted CBC-MAC: essentially if you instantiate our construction $\mathcal{F}(\mathcal{H})$ with CBC-MAC, you use F two times:

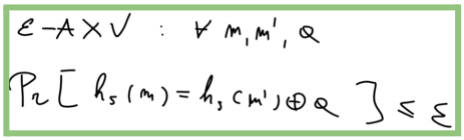
- to define the AU hash function (with the key s);
- after you have computed $h_s(m)$, you then do F_k of the output of the hash function.

What do we get in this case? The theorem that we have proven is that we get a FIL MAC (Thm 15) but actually we can prove that it gives a VIL MAC (we don't prove it).

XOR-MAC

XOR-MAC is based on other mode of operation. L'idea è invece di $F_k(h_s(\cdot))$ (what we did until now), utilizzare $(\eta, F_k(\eta) \oplus h_s(m))$ dove η è un random nonce. Essentially, in $F_k(h_s(\cdot))$ constructions we design input shrinking function matches the input size of F; in this other approach we design input shrinking function matches the output size of F (we xor the output). This construction gives only long input MAC and not long input PRF.

Observation. Dato un tag (η, v) per un messaggio m, l'avversario prova a forgiare $(\eta, v \oplus a)$ per $m \neq m'$. Quando è valido il tag? Sarà valido se $h_s(m) = h_s(m') \oplus a$. Nella costruzione precedente, l'unico modo per romperlo è trovare collisioni ($a = 0$); qui invece puoi fare di più di trovare collisioni, puoi provare ad aggiungere a ad ogni tag. Quindi le proprietà di AU non sono più sufficienti: abbiamo bisogno di definire ε-AXU (Almost XOR Universality).



(•) ε-AXU → It's hard to find a such that $h_s(m) = h_s(m') \oplus a$.

Da qui definiamo il teorema [Thm 16 - If $\mathcal{H}$ is ε-AXU and $\mathcal{F}$ is a PRF, then XOR-MAC is UF-CMA].

- Thm 16 - If $\mathcal{H}$ is ε-AXU and $\mathcal{F}$ is a PRF, then XOR-MAC is UF-CMA

MAC: FIL vs VIL

We essentially studied two approaches: the XOR construction and the $\mathcal{F}(\mathcal{H})$ and then we instanced those with different things like information-theoretically, inner product, polynomial evaluation or like CBC. But we only treat them as fixed input length. FIL (Fixed Input Length) means that Alice and Bob exchange the key and they always authenticate messages of the same length (fixed).

- FIL: fixed N or d (number of blocks fixed) arbitrary.
- VIL: N or d arbitrary but can change for each message.

	FIL-PRF	FIL-MAC	VIL-MAC
$\mathcal{F}(\mathcal{H})$	✓	✓	
CBC-MAC		✓	
E-CBC-MAC	✓	✓	✓
XOR-MAC		✓	✓

Which constructions are VIL secure? E-CBC and XOR-MAC. CBC is computationally AU (Lemma - CBC-MAC defines completely an AU family) and it is also a FIL MAC (Thm 15), but it is not VIL secure. How do you fix it? You fix it by encrypting (E-CBC).

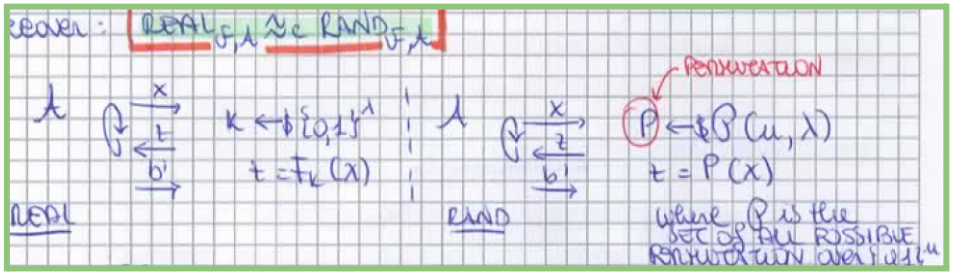
This concludes more or less what we will ever say about MAC.

PRP

Plan. Alice and Bob exchange the key, the key is short, they can use it forever and to encrypt messages of different length. We start with $Enc(k, m) = (r, F_k(r) \oplus m)$; what if $m = m_1, \ldots m_d$ (possibly VIL)? The answer is MODES OF OPERATION!

Alcuni modes richiedono di invertire F, ma la funzione non può essere invertita a priori. Possono essere invertite però le permutazioni: questo motiva la definizione di PRP.

What is a PRP? It's just a PRF that can be computed in both directions, forward and backward. $\mathcal{F} = \{F_k : \{0.1\}^n \to \{0.1\}^n\}_{k\epsilon\{0.1\}^\lambda}$ is a PRP [Def 19 - PRP] if $\forall k\epsilon\{0,1\}^\lambda$ exists a $F_k^{-1}(z)$ such that $F_k^{-1}(F_k(x)) = x$. Inoltre i due game real experiment (che utilizza le funzioni di $\mathcal{F}$) e rand experiment (che utilizza una funzione p computata randomicamente dal set di tutte le possibili permutazioni su $\{0.1\}^n$) sono computazionalmente indistinguibili.

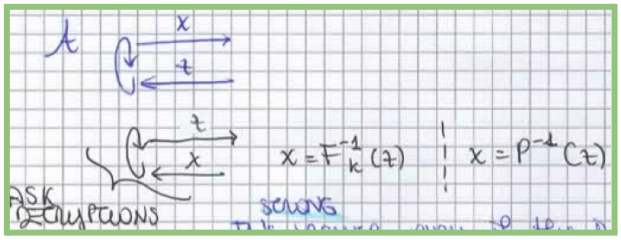


Come si costruisce la PRP? PLAN: costruire la nostra PRP.

- Def 19 - PRP

(CRYPTO 2022)
Definizione di PRP (Def 19) non fatta in maniera non formale.

AES è una STRONG PRP (strong, it is secure even if the inverse permutation is queried).



Lecture 10

[1]: 5.2, 5.3 [2]: 5.2 [7]: 3.5

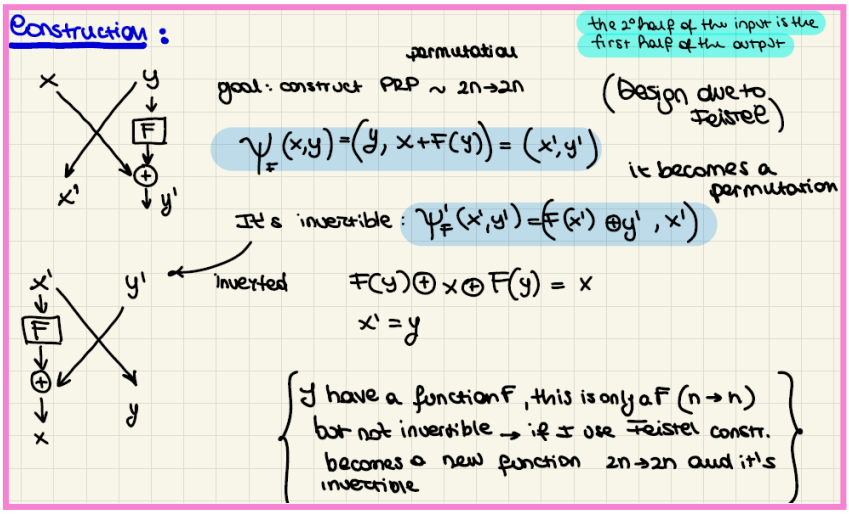
➢ Pseudorandom permutations (PRPs) and Feistel networks.

RECAP. Con PRF, possiamo costruire CPA-secure encryption e Message Authentication Codes. Nella scorsa lezione (Lecture 9), ci siamo occupati di estendere la lunghezza del nostro input per MAC; adesso vogliamo farlo anche per l'encryption. Per fare ciò abbiamo bisogno di invertire F: we need PRP.
La PRP corrisponde ad un block cipher: block ciphers are designed to be secure instantiations of (strong) pseudorandom permutations with some fixed key length and block length. How do we design efficient block ciphers?
Vogliamo ottenere confusion (ovvero il block cipher deve essere abbastanza complicato da far perdere un sacco di energie nel fare il reverse) e diffusion (ovvero che al minimo cambio cambi tutto il blocco). Everybody knows what the cipher does (public), only the key is secret.

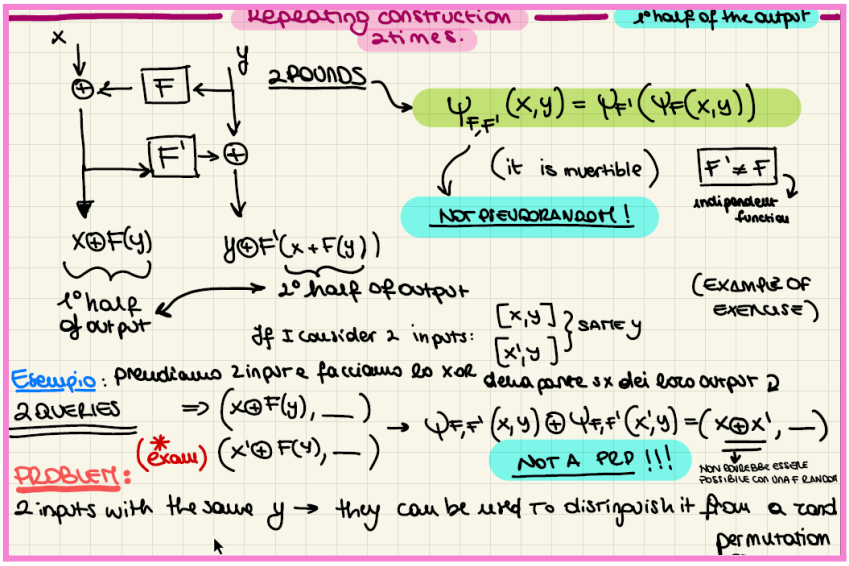
Ci sono vari esempi come AES (Lecture 11), DES, Camellia. Dopo aver stabilito come ci aspettiamo sia una PRP, il nostro PLAN è costruirla.

Feistel networks

Prendiamo una funzione F e creiamo una permutazione ψ_F in modo tale che questa possa essere invertita (ψ_F^{-1}) senza invertire la funzione stessa e preservando la sicurezza della funzione (vedi costruzione nell'immagine). QUESTION: Sia F quindi una PRF, ψ_F è una PRP? È sicura questa costruzione? No, perchè la seconda metà dell'input corrisponde alla prima metà dell'output, ovvero non abbiamo modificato y nell'output (not very pseudorandom). (•) Example. I ask for (x, y) and if I receive something that start with y I am in $Real$, if instead I receive something else I am in $Rand$.



Definiamo ora una Two-round-Feistel. QUESTION: $\psi_{F,F'}$ è una PRP? È sicura questa costruzione? No. Se prendessimo ad esempio due input (x, y) e (x', y) e facessimo lo xor della parte sinistra dei loro output (ovvero $x \oplus F(y)$ e $x' \oplus F(y)$), otterremmo $x \oplus x'$ (this should not be possible with a random function).



(•) Observation 1. To prove that it is not a PRP just prove that indeed there exists a distinguisher. I first query (x, y) and later I query (x', y); then i do the XOR:

- if I am in $Real$, I expect to see something that starts with $x \oplus x'$;
- if I am in $Rand$, this only happens with negligible probability.

So this gives us a distinguisher. At the exam, if you are asked for a similar exercise, you've to describe the distinguisher and you compute the probability that the distinguisher succeeds. So the problem why this is not a PRP is that if you query two inputs with the same y, they can be used to distinguish it from a random permutation!

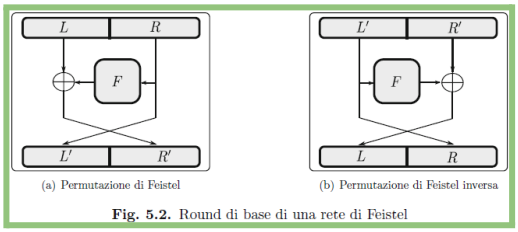
Exercise (+ idea). Prove Two-round-Feistel it's not a PRP. You show it exists a distinguisher of $Real$ and $Rand$ but given that $x \oplus x'$ it's negligible in $Rand$ then it's not a PRP.

Observation 2. If the sequence of input that the adversary ask don't share the same y, then this is a PRP! Of course in practice you cannot just assume that the adversary never makes a query with the same y, so somehow in the construction you need to enforce that this always happens.

Data quindi una PRF family $\mathcal{F}$, in generale possiamo scrivere $\psi_F[r] \equiv \psi_{F_{k1},\dots,F_{kr}}(x, y)$ con r che corrisponde al numero di round. Per ogni round abbiamo una differente chiave e quindi una differente F_k. How many rounds for ψ to be a PRP?

[1]: 5.2 (Reti di Feistel)

La costruzione seguente è detta permutazione di Feistel, in onore del suo creatore. L'input è costituito da una stringa di 2n bit, separata in due blocchi di n bit, nel seguito indicati con L ed R. Come mostrato in Fig. 5.2(a) la metà di destra (ovvero la stringa R) viene portata subito in uscita e costituisce la metà sinistra dell'output; d'altra parte il valore R è usato come input per la funzione F(·) e la stringa così ottenuta è sommata modulo 2 con il valore L, per ottenere la metà destra dell'output.



(a) Permutazione di Feistel (b) Permutazione di Feistel inversa

Fig. 5.2. Round di base di una rete di Feistel

Un po' più formalmente, data una funzione $F: \{\{0.1\}^n \to \{0.1\}^n\}$ possiamo costruire una permutazione $\Psi_F : \{0,1\}^{2n} \to \{0,1\}^{2n}$ come segue:

$$\Psi_F(L, R) = (R, F(R) \oplus L)$$
$$\Psi_F^{-1}(L, R) = (F(L) \oplus R, L).$$

Notare che anche ψ_F^{-1} è calcolabile efficientemente, come mostrato in Fig. 5.2(b). Una rete di Feistel è ottenuta iterando la permutazione di Feistel in diversi round; in generale ogni round farà uso di una diversa funzione $F_i(\cdot)$. (...) Come vedremo le reti di Feistel sono alla base di diverse costruzioni di cifrari a blocco usate in pratica, tra cui il DES (cf. Paragrafo 5.3). Il risultato di Luby e Rackoff è che, quando F(·) è una PRF (con o senza chiave), r = 3 round sono sufficienti per ottenere una PRP, ed r = 4 round generano una PRP forte.

[2]: 5.2 (Feistel Networks)

A Feistel network is an alternative approach for constructing a block cipher. The low-level building blocks (S-boxes, mixing permutations, and a key schedule) are the same; the difference is in the high-level design. The advantage of Feistel networks over substitution-permutation networks is that a Feistel network eliminates the requirement that S-boxes be invertible. This is important because a good block cipher should have "unstructured" behavior (so that it looks random); however, requiring that all the components of the construction be invertible inherently introduces structure. *A Feistel network is thus a way of constructing an invertible function from non-invertible components*. This seems like a contradiction in terms -- if you cannot invert the components, it seems impossible to invert the overall structure -- but the Feistel design ingeniously achieves this.

A Feistel network, as in the case of a substitution-permutation network, operates in a series of rounds. In each round, a round function is applied in a specific manner that will be described below; In a Feistel network, round functions need not be invertible. Round functions typically contain components like S-boxes and mixing permutations, but a Feistel network can deal with any round functions irrespective of their design. When the round functions are constructed from S-boxes, the designer has more freedom since the S-boxes need not be invertible.
The i-th round of a Feistel network operates as follows. The input to the round is divided into two halves denoted L_{i-1} and R_{i-1} (with L and R denoting the "left half" and "right half" of the input, respectively). If the block length of the cipher is n bits, then L_{i-1} and R_{i-1} each have length n/2, and the i-th round function f_i will take an n/2-bit input and produce an n/2-bit output. The output (L_i, R_i) of the round, where L_i and R_i again denote the left and right halves, is given by

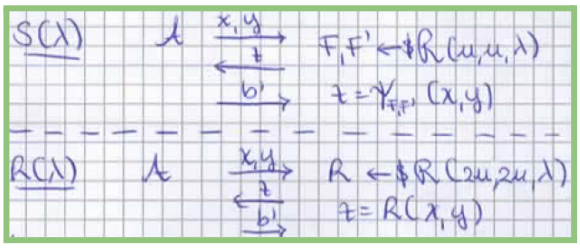
$$L_i := R_{i-1} \quad \text{and} \quad R_i := L_{i-1} \oplus f_i(R_{i-1}).$$

(•) Luby-Rackoff proved that three rounds are enough. We are gonna prove it now but essentially the intuition is that:

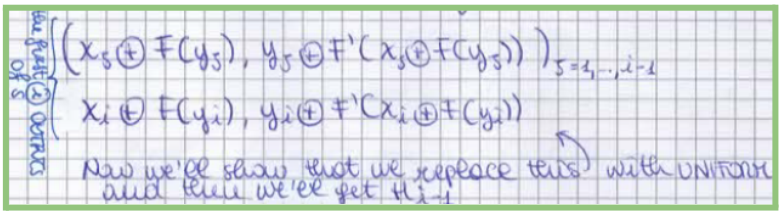
- if "y"s are different→ two rounds work;
- we add one round and this suffices to actually enforce.

I block cipher DES e AES sono realizzati come una Feistel Network in cui F e F' sono costruite ad hoc e non necessariamente sulla base di PRF e, il numero di round, non corrisponde necessariamente a tre.

Definiamo il teorema di Luby-Rackoff [Thm 17 - (Luby-Rackoff) If $\mathcal{F}$ is a PRF, $\psi_{\mathcal{F}}[3]$ is a PRP], per cui se $\mathcal{F}$ è una PRF, allora $\psi_{\mathcal{F}}[3]$ è una PRP.



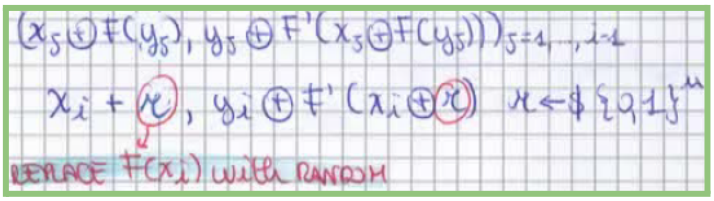
Proof strategy: two rounds are good so long "y"s are distinct; the third round makes sure "y"s are distinct. Definiamo un primo lemma [Lemma 1] che afferma che per tutti unbounded distinguishers, che effettuano $q(\lambda) = poly(\lambda)$ queries, $S(\lambda)$ e $R(\lambda)$ definite in figura sono indistinguibili (ovvero $SD(S(\lambda); R(\lambda)) \leq negl(\lambda)$) fintanto che tutte le queries sono uniche $((x_1, y_1), \dots, (x_q, y_q)\ s.t.\ \forall i \neq j, y_i \neq y_j$, ovvero "$y$"s are all distinct). Dimostriamo il lemma definendo un game ibrido $H_i(\lambda)$, che risponde alle prime i queries come $S(\lambda)$ e le successive come $R(\lambda)$: è sufficiente provare che $H_{i-1} \approx_s H_i\ \forall i = 1,\dots, q$. Look at the first i outputs of H_i:



For all other queries we have random: $R(x_j, y_j)$ for $j = i + 1, i + 2, \dots, q$.

In order to prove that $H_{i-1} \approx_s H_i$, I have to show that I can switch the i-th output with random. If I do that, I obtain H_{i-1} which means that H_{i-1} and H_i are indistinguishable. By uniqueness

of "y"s, $F(y_i)$ is random and independent of $F(y_1)$,..., $F(y_{i-1})$ as well as of all other outputs: then I can replace $F(y_i)$ with random.

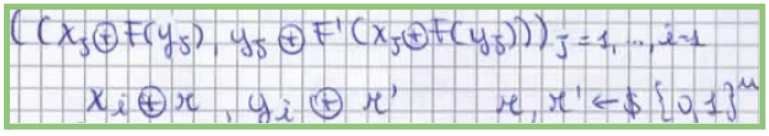


(•) If you want to prove that two distributions are indistinguishable you always need to consider the entire distribution: you cannot just look at the i-th element in isolation because the adversary guesses to see the entire sequence. So you make sure that the i-th element is indistinguishable from random given what the adversary knows, which are in this case the first $i - 1$ queries.

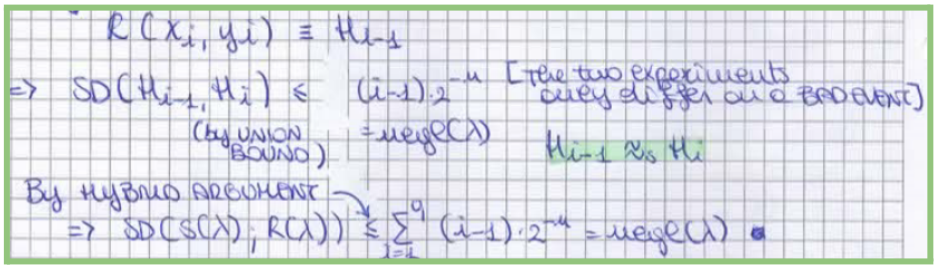
Now $x_i \oplus r$ is random and the probability that it will collide with the previous ones is negligible.

(•) What is the probability that $x_i \oplus r = x_j \oplus F(y_j)$ for $j < i$? For single j is 2^{-n}, because $F(y_j)$ is random and so for each $j < i$, by the union bound, it is $2^{-n}(i - 1)$.

Conditioning on this event not happening we get:



Now we can replace it as the output of the random function.



Procediamo quindi alla dimostrazione del teorema, considerando quattro esperimenti:

- $T(\lambda)$: $(x, y) \to \psi_{F_{k1}, F_{k2}, F_{k3}}(x, y)$ (REAL, Three-round Feistel)

  [(•) $T(\lambda)$: $(x, y) \to \psi_{F_{k3}}(\psi_{F_{k2}}(\psi_{F_{k1}}(x, y)))$, we pick three keys $k_1, k_2, k_3 \leftarrow \$U_\lambda$ and then we do Three-round Feistel with $F_{k1}, F_{k2}, F_{k3} \in \mathcal{F}$]
- $S(\lambda)$: $(x, y) \to \psi_{F,F'}(\psi_{F''}(x, y))$, $F, F', F'' \leftarrow \$\mathcal{R}(n, n, \lambda)$
- $R(\lambda)$: $(x, y) \to R(x, y), R \leftarrow \$\mathcal{R}(n, n, \lambda)$ ($R(\lambda)$ truly random function)
- $P(\lambda)$: $(x, y) \to P(x, y), P \leftarrow \$\mathcal{P}(2n, \lambda)$ (RAND, $P(\lambda)$ truly random permutation)

Definiamo quindi tre lemmi.

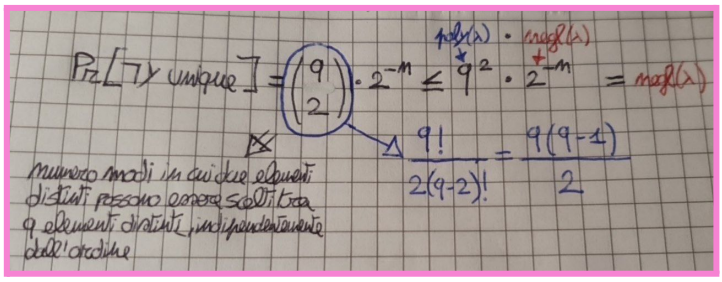
- [Lemma 2] $T(\lambda) \approx_c S(\lambda)$

Exercise (+ idea). Prove it. Viene provato utilizzando l'hybrid argument mediante PRF security e quindi switchando da T a S mediante una truly random function (hybrid argument: you first switch the first PRF, i.e. instead F_{k1} you put F'', then the first two PRFs and then all of them).

- [Lemma 3] $R(\lambda) \approx_s S(\lambda)$

Definiamo l'evento ***ynique*** (***ynique:*** $\psi_{F''}(x_i, y_i)$) e dimostriamo che la probabilità di questo è negligible (if this holds the lemma would be false).
(•) Usiamo il primo lemma (Lemma 1), che affermava che R e S sono statistically close fintanto che la sequenza di valori che l'avversario dava in output era ***ynique***. So all we need to prove is that for whatever adversary inputs (x, y), $\psi_{F''}(x, y)$ introduces a ***ynique*** sequence: just show that the probability that $\psi_{F''}(x, y)$ outputs a ***ynique*** sequence which is not ***ynique*** is negligible. If we can prove it, we have done, because we can invoke the first lemma.
Take any two queries (x_i, y_i) and (x_j, y_j) such that $(x_i, y_i) \neq (x_j, y_j)$. There are two cases:

1. if $y_i = y_j$, then $x_i \neq x_j \rightarrow$
   thus $y_i'' = x_i \oplus F''(y_i) = x_i \oplus F''(y_j) \neq x_j \oplus F''(y_j) = y_j'' \rightarrow y_i'' \neq y_j'' \rightarrow$ ***ynique***!
2. $y_i \neq y_j$
   (•) I don't get ***ynique*** if $x_i \oplus F''(y_i) = x_j \oplus F''(y_j) \rightarrow$
   which is the same of $x_i \oplus x_j = F''(y_i) \oplus F''(y_j) \rightarrow$ since y_i and y_j are different (because F'' is invoked on different values, so the outputs are different), $F''(y_i)$ and $F''(y_j)$ are both random $\rightarrow$ so $Pr[x_i \oplus x_j = F''(y_i) \oplus F''(y_j)] \leq 2^{-n}$. This is for a fixed pair of i, j so if I want that the entire sequence is ***ynique***, this need to be for all pairs…



- [Lemma 4] $R(\lambda) \approx_s P(\lambda)$.

(•) What is the difference between R and P? P is a permutation (truly random permutation), R is a function (truly random function), thus P can be inverted. So in terms of truth table we have a function and not a permutation if it cannot be inverted and this happens if there are two inputs that go to the same output (because it is a one to one map). The only way to distinguish between R and P is if you are able to find two values that go to the same output (collisions). This happens with negligible probability: $Pr[\exists i, j: R(x_i, y_i) = R(x_j, y_j)] \leq q^2 * 2^{-n}$.
Quindi proviamo il teorema $T \approx_c S \approx_s R \approx_s P$.

- Thm 17 - (Luby-Rackoff) If $\mathcal{F}$ is a PRF, $\psi_{\mathcal{F}}[3]$ is a PRP (+ proof) + Lemma 1 (+ proof) + Lemma 2 (+ proof as Exercise) + Lemma 3 (+ proof) + Lemma 4 (+ proof)

(CRYPTO 2022)
Nel 2021, Lemma 4 lasciato come esercizio; nel 2022, Lemma 4 con dimostrazione.
Nel 2022, Thm 18 non nominato.

È possibile inoltre dimostrare (we won't prove this) che $\psi_{\mathcal{F}}[4]$ è una strong PRP [Thm 18 - $\psi_{\mathcal{F}}[4]$ is a strong PRP].

- Thm 18 - $\psi_{\mathcal{F}}[4]$ is a strong PRP

Lecture 11

[1]: 5.4 [2]: 3.6, 3.7, 4.5 [7]: 3.4

- ➢ Modes of operation: ECB, CBC and CTR.
- ➢ Proof that CTR mode using a PRF yields a CPA secure SKE for variable length messages.
- ➢ Definition of chosen-ciphertext attack (CCA) secure SKE.
- ➢ Authenticated encryption and its relation to CCA security.

RECAP. Due cose ci sono rimaste da fare:
1. Domain extension for SKE (How do we encrypt long messages?);
2. Authenticated encryption (how authentication+encryption?).

Domain extension for SKE

(•) We know that in theory we can construct SKE having OWF. In the real world:

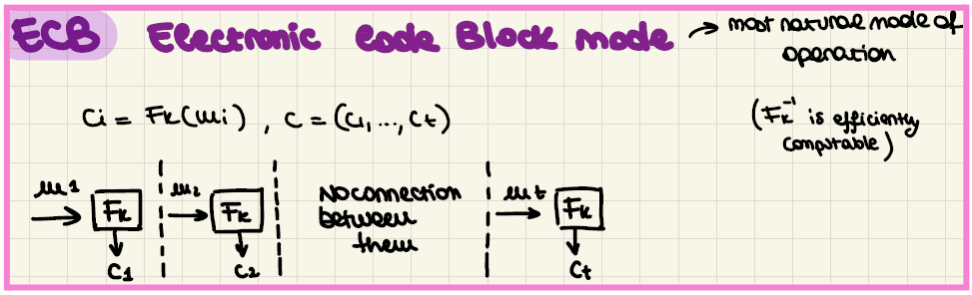
1. Either get PRF for concrete assumptions (Factoring, we will see later because we need number theory);
2. Heuristic construction (AES).

AES (Advanced Encryption Standard) is actually a PRP. At the moment, we can think of AES as the practical mode to get PRF (actually it is more, we have permutation → PRP). AES property is pseudorandomness, but it's not a block cipher (not encryption). It encrypts a block, as a block cipher, but it's a deterministic function so it's not secure. AES is a NIST Standardization process (introduced after DES). Still no one has broken it but if someone does it, it would not be a surprise. During the course we're gonna to talk about concrete assumptions to do public encryption (number theoretical) and how to design a PRP, that is the theoretical justification of AES construction. It's like a mixing of experience (we want to be fast through practice) and theory (there is something theoretical). But before that, we need to introduce VIL for AES (that is a PRF). It's like we want to allow that the length of the message exchanged between Alice and Bob can be altered. This is called modes of operation. Are these modes secure? We're going to prove it.

Modes of operation: ECB, CFB, CBC and CTR

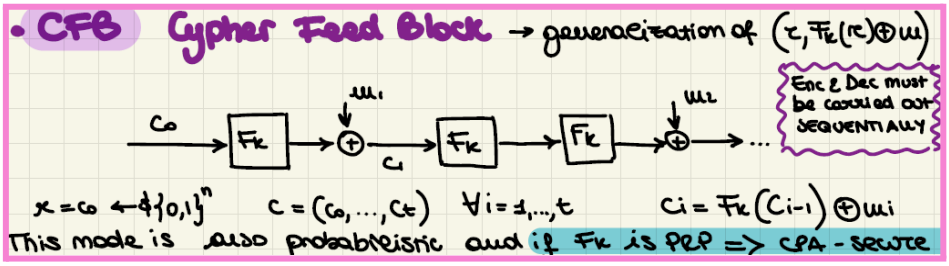
Modes of operation are typically defined along the block ciphers. Dato un block cipher con fixed input length (e.g. AES), come facciamo l'encryption di lunghi messaggi $m = (m_1, \ldots, m_t)$ with $m_i \epsilon \{0,1\}^n$? Noi faremo riferimento sulla PRF (PRP) $\mathcal{F} = \{F_k\}$ (AES è una PRF).

- ECB MODE (Electronic Code Book Mode)



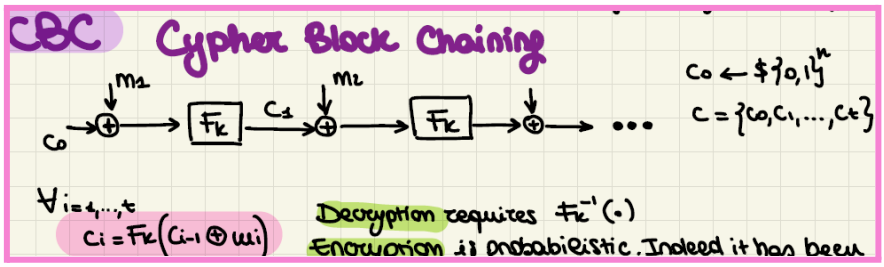
La nostra prima idea è ECB MODE, the most natural mode of operation. La decryption viene fatta sfruttando il fatto che F_k^{-1} è efficientemente computabile. ECB MODE non è sicuro (non è una secure encryption) perché è deterministico e quindi viola la CPA secure (no deterministic encryption can be CPA secure, vedi Def 15). Non cambia questa cosa con la sicurezza di PRF. AES (PRF) deve essere usata con un certo mode of operation per avere encryption, ma non ci dà da solo encryption.
(•) Don't use it! If the same block is repeated twice in the plaintext, this can be detected as a repeating block in the ciphertext; thus, it is easy to distinguish an encryption of a plaintext that consists of two identical blocks from an encryption of a plaintext that consists of two different blocks.

- CFB MODE (Ciphertext Feedback Mode)



Richiamando la nostra costruzione $c = (c_1, c_2) = (r, F_k(r) \oplus m)$ (vedi Construction di Thm 11), abbiamo che $c_i = F_k(c_{i-1}) \oplus m_i$. F_k non ha bisogno di essere invertita per invertire l'encryption. Adesso l'encryption è diventata sequential ma è ancora sicura: questa construction è CPA secure.
(•) It is the generalization of $(r, F_k(r) \oplus m)$. This mode is probabilistic and if F_k is PRP, CFB MODE encryption is CPA secure (we won't prove this). Decryption doesn't require F_k^{-1}. Here both encryption and decryption must be carried out sequentially.

- CBC MODE (Cipher Block Chaining Mode)

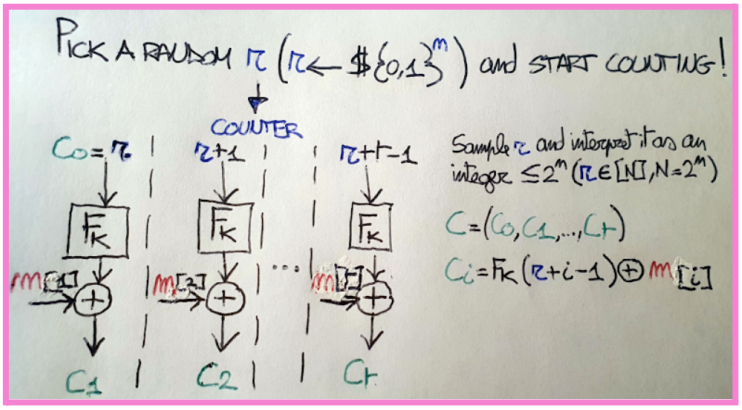


CBC MODE è CPA secure (we won't prove it). Per invertire questo (decryption), abbiamo bisogno di invertire F_k (we need PRP).

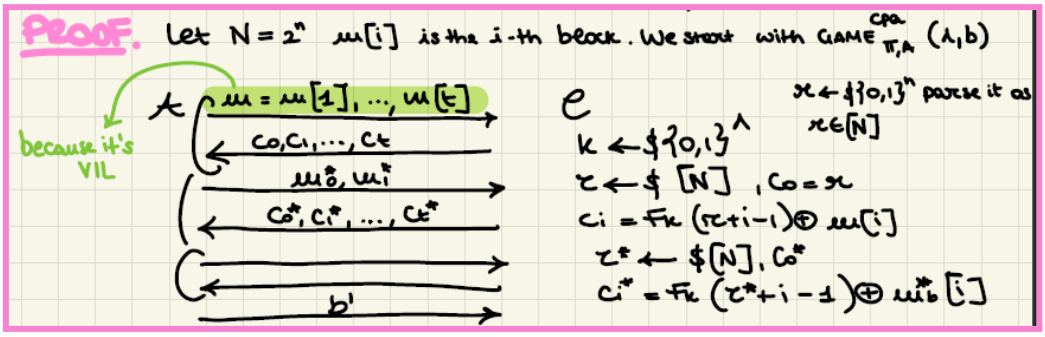
(•) Encryption in CBC MODE is probabilistic. Indeed it has been proven that if F_k is PRP → CBC MODE encryption is CPA secure. The main drawback of this mode is that encrypting must be carried out sequentially because the ciphertext block c_i is needed in order to encrypt the plaintext block m_{i+1} (unlike decrypting, which may be executed in parallel). Thus, if parallel processing is available, CBC MODE encryption may not be the best choice.
CBC MODE è collegato a CBC-MAC, ma sono differenti:

- CBC MODE is for encryption, so we also add c_o at the beginning that needs to be random which is xored to x_1, while CBC-MAC doesn't use c_0 because doesn't need any randomness;
- in CBC-MAC non si fa l'output di tutti i blocchi perchè tu hai bisogno solo di autenticare il messaggio (non hai bisogno di recuperarlo), mentre in CBC MODE devi farlo perchè hai bisogno di decriptare il messaggio.

- CTR MODE



È la costruzione migliore perché, oltre a garantire security, ha random access [(•) you can encrypt or decrypt the block that you want, you don't need the previous ones], parallelizable encryption e parallelizable decryption. Definiamo quindi il teorema [Thm 19 - Assuming $\mathcal{F}$ is a PRF family, then CTR mode yields a CPA-secure VIL SKE] per cui CTR MODE è CPA secure per VIL, se $\mathcal{F}$ è una PRF. Per la dimostrazione, modelliamo un CPA attack per un CTR MODE cipher. Definiamo quindi il $Game^{CPA}_{\pi, A}(\lambda, b)$ e dimostriamo che $Game^{CPA}_{\pi, A}(\lambda, 0) \approx_c Game^{CPA}_{\pi, A}(\lambda, 1)$. (•) Note. We wrote always t, but actually the j-th query has t_j number of blocks and challenges query has t^* number of blocks.



Per provarlo, utilizziamo due esperimenti:

- $Hyb_1(\lambda, b)$, where we replace the function F_k with the random function R, that is deterministic function but computed with random truth table ($R \leftarrow \$\mathcal{R}(n, n, \lambda)$);
- $Hyb_2(\lambda)$, that it is as $Hyb_1(\lambda, b)$ but c^* is uniform (random ciphertext).

Definiamo quindi due lemmi.

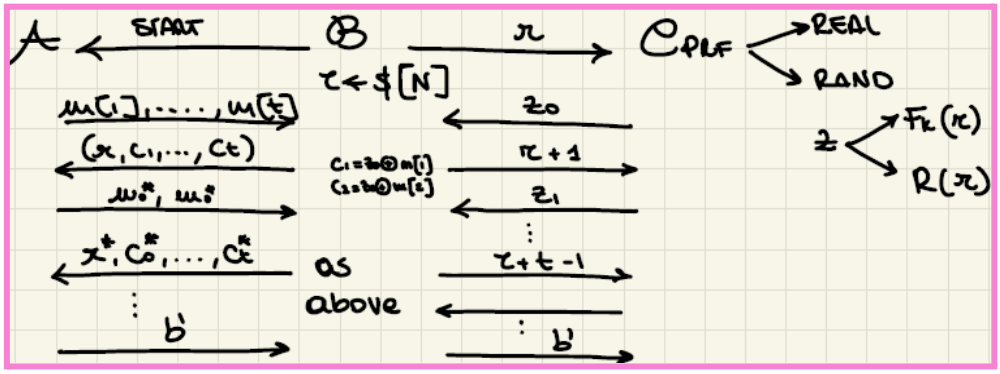
- [Lemma 1] $Game^{CPA}_{\pi, A}(\lambda, b) \approx_c Hyb_1(\lambda, b)\ \forall b \epsilon \{0, 1\}$

(•) We prove the first lemma by reduction. Assume lemma is not true…

→ then $\exists PPT\ A\ s.t\ |Pr[Game^{CPA}_{\pi, A}(\lambda, b) = 1] - Pr[Hyb_1(\lambda, b) = 1]| \geq 1/poly(\lambda)$ for some b

→ then $\exists PPT\ B\ s.t\ |Pr[Real_{\mathcal{F},B}(\lambda) = 1] - Pr[Rand_{\mathcal{F},B}(\lambda) = 1]| \geq 1/poly(\lambda)$

(i.e B breaks PRF). So we can proceed to construct B.



Sequence → B starts A. A sends $m[1],..., m[t]$. Then B takes $r \leftarrow \$[N]$ and sends to C. C sends back z_0 that if $Real$ is equal to $F_k(r)$, if $Rand$ is equal to $R(r)$. Then B sends $r + 1$ to C and receives z_1 and so on up to $r + t - 1$ and B receives z_t. Then B sends to A $(r, c_1,..., c_t)$ where $c_i = z_{i-1} \oplus m[i]$ for $i = 1,..., t$. Then A sends m_0^*, m_1^*. B picks $r^* \leftarrow \$[N]$ and do as above. And so on.

By inspection, when B plays in $Real$ the view of A is identical to the view of $Game$, while when B plays in $Rand$ the view of A is identical to the view of Hyb_1:

$Pr[Real_{\mathcal{F},B}(\lambda) = 1] = Pr[Game_{\Pi,A}(\lambda, b) = 1]$ (the probability that B outputs 1 in $Real$ is equal to the probability that A outputs 1 in $Game$)

$Pr[Rand_{\mathcal{F},B}(\lambda) = 1] = Pr[Hyb_1(\lambda, b) = 1]$

By assumption:

$|Pr[Real_{G,B}(\lambda) = 1] - Pr[Rand_{G,B}(\lambda) = 1]| =$

$|Pr[Game_{\Pi,A}(\lambda, b) = 1] - Pr[Hyb_1(\lambda, b) = 1]| \geq 1/poly(\lambda)$. (contradiction!)

B using A can distinguish and so it breaks PRF. This proves the lemma.

- [Lemma 2] $Hyb_1(\lambda, b) \approx_s Hyb_2(\lambda)\ \forall b$

(•) In Hyb_1, for the challenge we pick a random r^*, we start counting and we put the counter as input of R (we do the same for the encryption queries with r_j):

- $R(r^*), R(r^* + 1), \ldots, R(r^* + z^* - 1)$
- $R(r_j), R(r_j + 1), \ldots, R(r_j + z_j - 1)$.

These are the sequences of values that I xor with the messages. I can replace the challenge ciphertext with the uniform random string (Hyb_2) when I do like OTP, that means that the

values that I xor with the message blocks needs to be random and never seen before and so we want that the sequences of input of R are always fresh. In particular, if it happens that the sequence $r^*, r^* + 1, \dots, r^* + z^* - 1$ doesn't overlap with any of other sequences, it means that in the challenge step we are evaluating R on fresh input and so we are taking a fresh raw never seen before: the challenge ciphertext is gonna be random. Instead if there is an overlap, I can distinguish Hyb_1 from Hyb_2. So let *BAD* be the event that $\exists i, j, j' \geq 1$ such that $r_j + i = r^* + j'$ (i.e. at some point overlap); when *BAD* doesn't happen, the sequence $r^*, r^* + 1, \dots, r^* + z^* - 1$ does not overlap and then $c_0^*, c_1^*, \dots, c_{z^*}^*$ is uniform as in Hyb_2:

$\forall A, \forall b, |Pr[Hyb_1(\lambda, b) = 1] - Pr[Hyb_2(\lambda, b) = 1]| \leq Pr[BAD]$.

Now we need to bound $Pr[BAD]$:

(Note: for simplicity $j = t_j = t^* = q(\lambda) = poly(\lambda)$, i.e. every query is made of exactly q blocks; the number of queries is different from the length of each query and this can be different from the length of the challenge and so we take for simplicity the max length)

$Pr[BAD] = Pr[\exists j \, s.t. \, BAD_j] \leq \Sigma_j Pr[BAD_j] \rightarrow$

if $r^* - q + 1 \leq r_j \leq r^* + q - 1$ happens, then BAD_j happens $\rightarrow$

$\Sigma_j \frac{2q(\lambda)-1}{2^n}$ are all the cases that make the event happen, all values in the interval

$[r^* - q + 1, r^* + q - 1]$ that are $(r^* + q - 1) - (r^* - q + 1) + 1 \rightarrow$

So $\Sigma_j Pr[BAD_j] \leq \Sigma_j \frac{2q(\lambda)-1}{2^n} = \frac{2q(\lambda)-1}{2^n} * q = poly(\lambda) * negl(\lambda) = negl(\lambda)$

Possiamo quindi provare il teorema:

$Game(\lambda, 0) \approx_c Hyb_1(\lambda, 0) \approx_s Hyb_2(\lambda) \approx_s Hyb_1(\lambda, 1) \approx_c Game(\lambda, 1)$.

- Thm 19 - Assuming $\mathcal{F}$ is a PRF family, then CTR mode yields a CPA-secure VIL SKE (+ proof) + Lemma 1 (+ proof) + Lemma 2 (+ proof)

(CRYPTO 2022)

Sopra presentata l'intera dimostrazione del 2022 del Thm 19.

Nel 2022, dimostrazione completa del Lemma 1 (nel 2021, Exercise).

Nel 2022, esplicitato come lemma il fatto che Hyb' non dipende dal bit.

[Lemma] $Hyb_2(\lambda, 0) \equiv Hyb_2(\lambda, 1)$

Proof. Because Hyb_2 doesn't depend on b.

---

[1]: 5.4 (Modi operativi)

Un cifrario a blocco protegge blocchi di testo a lunghezza fissa (e.g., 64 bit per il DES e 128 bit per AES). Siccome in generale un messaggio da cifrare può avere lunghezza arbitraria, è necessario specificare un modo operativo: si divide il testo in chiaro in blocchi e si individua una modalità per proteggere i diversi blocchi così ottenuti attraverso il cifrario a blocco sottostante. (...)

Nel seguito supporremo che il messaggio m sia costituito da B blocchi, ovvero m = m1 || . . . || mB; assumeremo inoltre che la dimensione di ciascun blocco sia compatibile con la dimensione dell'input del cifrario a blocco sottostante, che indicheremo con la funzione P(·) (modellata come una PRP).

Modalità ECB. Il modo operativo più naturale è detto ECB (dall'inglese Electronic Code Book). La cifratura del messaggio m è ottenuta cifrando ciascun blocco indipendentemente, ottenendo così c =

Fk(m1) || . . . || Fk(mB). La decifratura è eseguita in modo ovvio, sfruttando il fatto che $F_k^{-1}(\cdot)$ è calcolabile in modo efficiente. Questo modo operativo è deterministico e quindi esso non può mai essere CPA-sicuro. In effetti tale modo operativo non è nemmeno semanticamente sicuro, in quanto due blocchi identici di testo in chiaro sono mappati in due blocchi identici nel testo cifrato: è banale dunque distinguere la cifratura di un messaggio m0 costituito da due blocchi identici, dalla cifratura di un messaggio m1 costituito da due blocchi distinti.

Modalità CBC. Nel modo operativo CBC (dall'inglese Cipher Block Chaining), si sceglie un vettore iniziale a caso IV della dimensione di un singolo blocco. Quindi ciascun blocco del testo cifrato è ottenuto applicando la funzione Fk(·) alla somma modulo due tra il blocco corrente del testo in chiaro ed il blocco precedente del testo cifrato. In simboli $c_i = F_k(c_{i-1} \oplus m_i) \qquad \forall\, i = 1, \ldots, B,$ dove c_0 = IV. Come vedremo questo modo operativo è CPA-sicuro. Lo svantaggio principale è che la cifratura deve essere eseguita in modo sequenziale, ovvero è necessario attendere che c_{i-1} sia generato prima di cifrare m_i. Osserviamo inoltre che la propagazione di un errore in un blocco di testo cifrato è limitata a due blocchi contigui in decifratura.

Modalità CFB. (...) questo modo operativo è CPA-sicuro. Come per il modo operativo CBC, la cifratura non è eseguibile in parallelo ed un errore in un blocco di testo cifrato affligge due blocchi in decifratura.

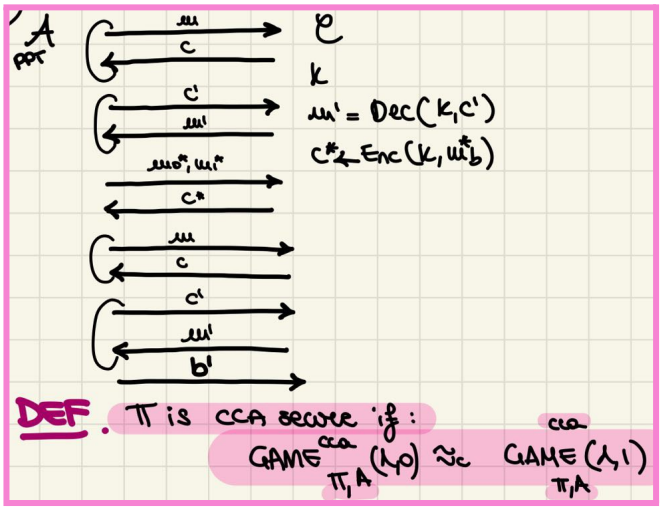
Modalità contatore. Nel modo operativo CTR (dall'inglese counter, contatore) si utilizza un contatore per generare un flusso pseudocasuale. (...) Anche questo modo operativo è CPA-sicuro. Sia la cifratura che la decifratura possono essere eseguite in parallelo ed il flusso pseudocasuale può essere pre-calcolato, dato che è indipendente dal messaggio da cifrare. Osserviamo inoltre che è possibile decifrare un singolo blocco di testo cifrato indipendentemente da tutti gli altri.

Authenticated encryption and its relation to CCA security

We want: (i) hide the message (confidentiality) and (ii) authenticate the message (integrity) and its source. How do we do message confidentiality and integrity? CCA security.

CCA security

Definiamo lo standard CCA (Chosen Ciphertext Attacks) security [Def 20 - CCA-security]. A differenza di CPA (Def 15) in cui l'avversario può conoscere qualcosa nel processo di encryption, in CCA l'avversario può conoscere qualcosa sul processo di decryption: ciò rappresenta la possibilità che, nella realtà, un avversario può imparare il testo in chiaro corrispondente ad alcuni testi cifrati ([1]: 5.1 (Sicurezza contro attacchi a crittotesto scelto)). In pratica l'attaccante, oltre a richiedere di vedere la cifratura $c_i = Enc(k, m_i)$ per messaggi scelti m_i (CPA), può anche inviare testi cifrati c_i' al challenger, ottenendo come risposta il corrispondente testo in chiaro $m_i' = Dec(k, c_i')$. Sebbene l'avversario mantenga la possibilità di interrogare sia prima che dopo aver scelto i due messaggi m_0^*, m_1^* (sulla base delle risposte ricevute fino a quel momento), ovviamente l'avversario non può interrogare il challenger in corrispondenza del crittotesto sfida c^*



(altrimenti otterrebbe proprio uno dei due messaggi e potrebbe facilmente distinguere la cifratura all'interno del game in figura, indicando il bit b' corrispondente). Ciò (il fatto che ***the adversary cannot query*** c^*) corrisponde nella vita reale al fatto che l'avversario non può fare attualmente decryption queries e ottenere il plaintext.

(•) These are two games one, where the challenger always encrypts m_0^* ($Game_{\Pi,A}^{CCA}(\lambda, 0)$) and one where the challenger always encrypts m_1^* ($Game_{\Pi,A}^{CCA}(\lambda, 1)$): these two games should be hard to distinguish.

- Def 20 - CCA-security

(•) CCA has two important meanings in real life.

- TLS

The first one is for the TLS protocol. At the beginning no CCA: the message gets padded. In this sense, it was performed a PKE (public key encryption) and decryption to check correctness of the message. What could happen? The server could have an oracle and the adversary may change the ciphertext to see how the server reacts. This is the so-called chosen ciphertext attack. For this reason CCA has been adopted.

- Non-malleability

The second real meaning is for non-malleability. This is a property that can be attacked in this way. Let's consider a message m, a ciphertext c, an adversary A and a ciphertext $c' \neq c$. The message is inside the ciphertext. Then the adversary starts to encrypt m and then it performs the attack such that m' is inside c' with $m' = 2m$ (the attacker knows this equality). This is the so-called malleability attack. This is a problem for instance for digital auctions. Assume that you make a digital auction, something like ebay. I want to buy something and I send my bid encrypted because I don't want the other to see it. Now you get my bid and, because this is CPA secure, you cannot see what is in the ciphertext. But then if it is malleable, it is not CCA secure and you don't care about what is inside, you just mall it in a way that your bid becomes twice my bid: then you are sure you win.

So CCA security implies a property called malleability: if you change the ciphertext a bit you don't get similar messages ([7]). A malleable scheme can never be CCA-secure.
(•) Malleable encryption can be good or not for some real applications.

(•) Exercise (+ solution). Let $Enc(k, m) = (r, F_k(r) \oplus m)$ be a CPA secure encryption based on PRF, prove that it is not CCA secure. The idea is that this encryption is malleable, because the second part is just a xor with the message. And if you flips a bit in the second part, this essentially corresponds to flip a bit in m. Whenever something is malleable, it is not CCA secure! Simple attack: $m_0^* = 0^n$; $m_1^* = 1^n$; then I receive $c^* = (r^*, s^*)$ where $s^* = F_k(r^*) \oplus m_b^*$ → I compute $s' = s^* \oplus 1||0^{n-1}$ (flip the first bit), so $c' = (r^*, s') \neq c^*$ → $m' = Dec(k, (r^*, s')) = F_k(r^*) \oplus s' = F_k(r^*) \oplus F_k(r^*) \oplus m_b^* \oplus 1||0^{n-1} = m_b^* \oplus 1||0^{n-1}$ (so what I get is the message with the first bit flipped) →

If $m' = 1||0^{n-1}$ then I output $b' = 0$ (because the message encrypted was m_0^*);

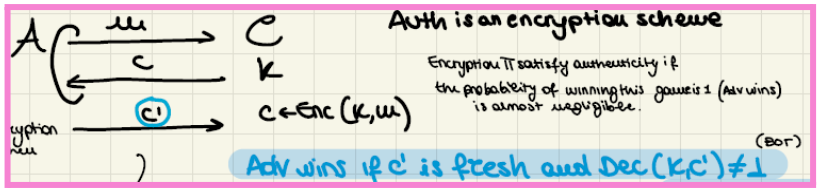
else $m' = 0||1^{n-1}$ then I output $b' = 1$.

Authenticity

Otteniamo CCA da due criteria (properties):

1. CPA (Def 15)
2. Authenticity (hard to produce a VALID ciphertext without knowing key, Def 21)

La proprietà di authenticity ci dice quindi che senza la chiave, possono essere prodotti solo ciphertext invalidi. Introduciamo il simbolo ⊥ per indicare la non validità di un ciphertext. Uno schema Π soddisfa per definizione la proprietà di authenticity se la probabilità di produrre un ciphertext valido senza conoscere la chiave è al massimo negligible [Def 21 - Authenticity].

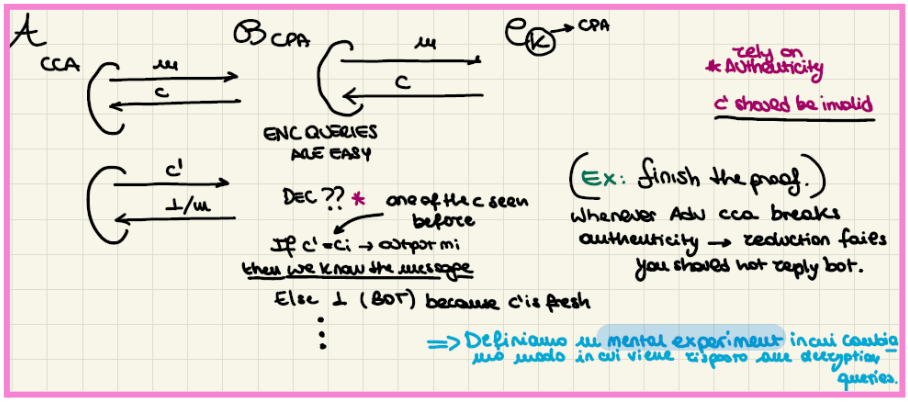


- Def 21 - Authenticity

Dimostriamo quindi che se un encryption scheme soddisfa le proprietà CPA e Auth, allora questo sarà CCA secure [Thm 20 - CPA + Auth ⇒ CCA].

Exercise (+ idea). Prove the theorem. Possiamo iniziare procedendo come prima idea con la riduzione da CCA a CPA: notiamo che questo metodo è problematico, poiché, a differenza delle encryption queries, le decryption queries non possono essere inoltrate a C_{CPA} (che non fa decryption). Per la proprietà della authenticity, non possiamo creare nuovi ciphertext (ciphertext is invalid unless it's old, not fresh). Abbiamo quindi che:

- if la decrypt query è uno dei ciphertext precedentemente ritornati dalle encryption queries → ritorna il messaggio;
- else (se è differente) → ritorna ⊥.



(•) We need to compute the probability that this adversary wins and in order to do that we need to consider some *BAD* event: *BAD* → the adversary breaks auth. Of course whenever A^{CCA} breaks auth, the reduction fails because you reply ⊥ when you shouldn't; but assuming that he never breaks it, the reduction is perfect. The "problem" of this reduction is that it is not so simple to analyze because we are mixing up two properties: CPA security and authenticity.

Another approach is to separate the proof in two steps.

1. You define an hybrid game that is equal to CCA game, only that you abort whenever the adversary makes a decryption query that violates the authenticity (you answer encryption queries normally, but decryption queries like the reduction we have done before). So $Hyb(\lambda, 0)$ is indistinguishable from $Hyb(\lambda, 1)$.

2. Prove that the original game is indistinguishable from Hyb (the only way to distinguish these is to make a query that breaks authenticity, so by auth they are indistinguishable).

Separiamo quindi la dimostrazione per le due proprietà (two Hyb and two Reductions, one for CPA one for AUTH).

- Thm 20 - CPA + Auth ⇒ CCA (+ proof as Exercise)

Combining SKE and MAC schemes

Three approaches used in real life to combine CPA security with MACs

a. ENCRYPT-and-TAG

$c \leftarrow \$Enc(k_1, m);\ \tau = Tag(k_2, m);\ c^* = (c, \tau)$

We use independent keys for different primitives (k_1 for CPA, k_2 for MAC). We are doing a BLACK BOX construction (have Enc and Tag and suppose they are secure). La costruzione è sicura? No, non è CPA secure (quindi di conseguenza non è CCA): Enc è CPA secure e Tag è UF-CMA secure, ma la combinazione non è sicura; HINT: Tag non è CPA secure.
(•) Exercise (similar to exam, + solution). Prove that this is not CCA secure. In order to solve such exercise we need to give a combination of Enc and Tag for which the construction is not secure (prove it's not secure → prove exists a *BAD* combination). τ could reveal something about the message, so let's find a tag that is unforgeable, but also reveal something about the message. We start with Tag which is UF-CMA, from this we define Tag' → $Tag'(k, m) = Tag(k, m)||m[1]$ (Note: $m[1]$ is the first bit of the message). Tag' is UF-CMA, but it reveals the first bit of the message! So using this Tag', the construction is not CPA secure (and not CCA).

b. TAG-then-ENCRYPT

$\tau = Tag(k_2, m);\ c \leftarrow \$Enc(k_1, m||\tau);\ c^* = c$

I don't output τ, otherwise i have problems as before. The construction is not CCA secure (it exists a bad combination, not always secure).
(•) In real world, TLS uses this to create the channel after the exchange of secret keys (authenticate then encrypt). The combination used in TLS can be proved that is secure.

c. ENCRYPT-then-TAG

$c \leftarrow \$Enc(k_1, m);\ \tau = Tag(k_2, c);\ c^* = (c, \tau)$

Questa costruzione funziona sempre (proof all'inizio di Lecture 12).

Lecture 12

[1]: 7.4, 4.1 [2]: 4.5, 5.1 [7]: 3.4, 3.6

- ➢ Combining confidentiality and message integrity: Encrypt-then-MAC and proof of CCA security.
- ➢ Collision-resistant hash functions.

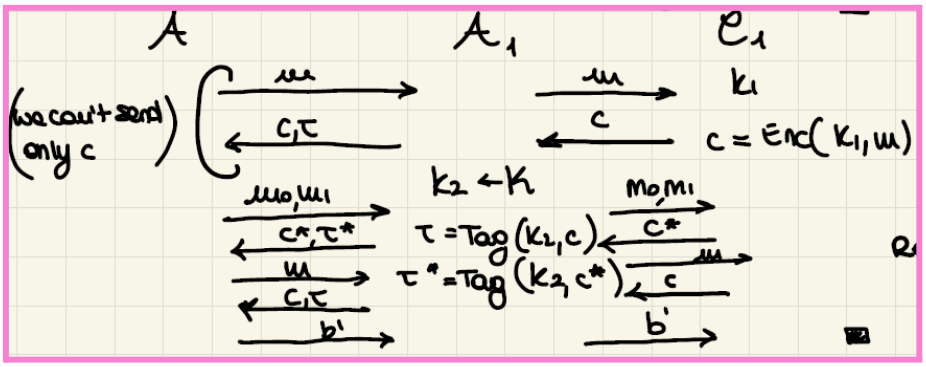
Encrypt-then-MAC and proof of CCA security

Dimostriamo quindi che l'approccio ***c*** (ENCRYPT-then-TAG, fine Lecture 11) è CCA secure [Thm 21 - Encrypt-then-MAC is CCA secure, assuming Π_1 is CPA-secure and Π_2 is

STRONG UF-CMA]. Let's call Π the combined scheme, the one that output $c' = (c, \tau)$. By previous theorem (Thm 20) suffices to show that Π (1) is CPA secure and (2) that satisfies authenticity.

1. CPA security

We prove the security by reduction: assume that Π is not CPA secure, so there exists an adversary A that with probability at least $1/poly(\lambda)$ can break the CPA security; then we construct a PPT adversary A_1 that breaks CPA security of Π_1.

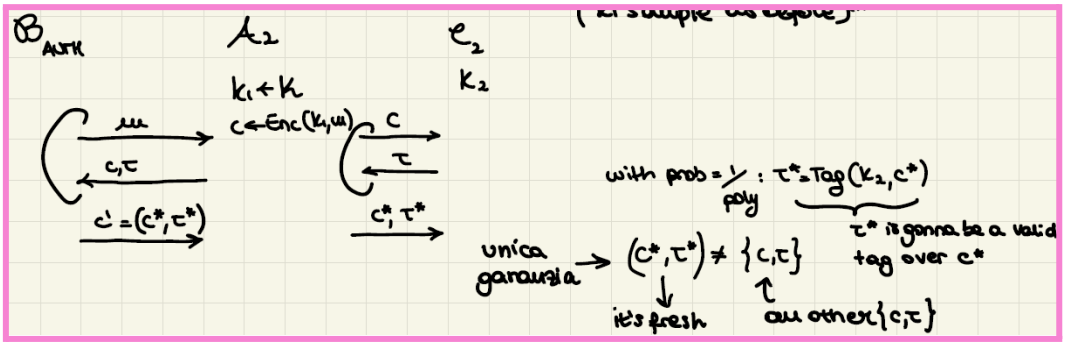


Sequence → A sends m to A_1. A_1 sends m to C_1. Then A_1 has to send (c, τ) because A is an adversary for Π: so to compute τ, he picks his own k_2. A_1 attacks the key k_1 which of course he doesn't know. But the reduction can sample it.

The simulation is perfect! So the probability that A' can distinguish is the same to the probability that A can distinguish that is at least $1/poly(\lambda)$ (contradiction!).

2. Authenticity

We prove it by reduction: assume that Π is not auth, so there exists PPT adversary B that with probability at least $1/poly(\lambda)$ breaks auth; then we can construct a PPT adversary A_2 that breaks the UF-CMA of Π_2.



Sequence → The challenger knows k_2. A_2 can sample k_1. B sends m (encryption query). A_2 encrypts it and sends it to C_2; then he receives τ and sends back (c, τ) to B. Now B is gonna forge on a ciphertext. he's gonna create a ciphertext c'.

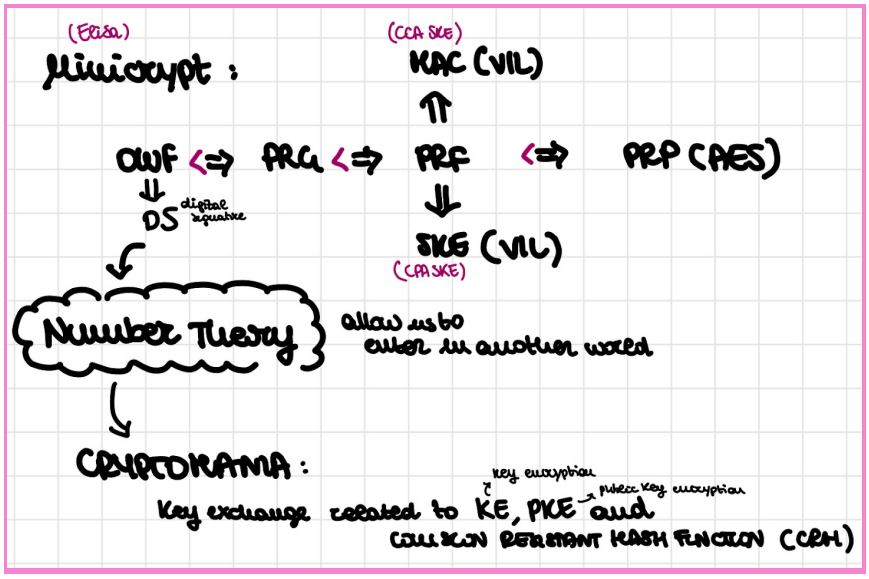
The simulation is perfect! Therefore, since B also breaks auth, this c' has the property that (c^*, τ^*) is fresh and τ^* is valid, which means $\tau^* = Tag(k_2, c^*)$ with probability at least $1/poly(\lambda)$. A_2 wins UF-CMA when $c^* \neq \{c\}$ (c^* is fresh) and $\tau^* = Tag(k_2, c^*)$ (τ^* is a valid tag on c^*). The crucial point is that c^* needs to be different from all the other c. But unfortunately this is not true, because the only condition that we have is that $(c^*, \tau^*) \neq \{c, \tau\}$, only the pair

is fresh it could be that c^* is equal to some previous c as long as the tag is fresh. So A_2 doesn't win UF-CMA, but breaks STRONG UF-CMA, because in this case it doesn't matter that c^* needs to be fresh because even if $c^* \neq c$ for some query, as long as τ^* is fresh. We have broken STRONG UF-CMA.

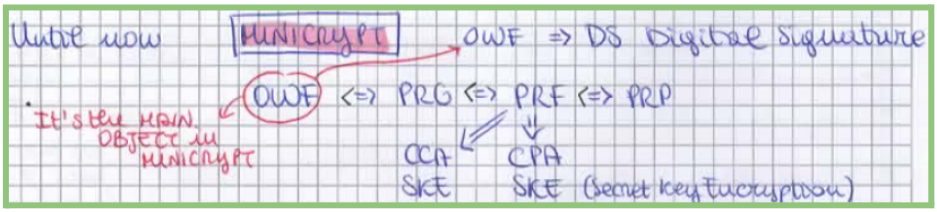
- Thm 21 - Encrypt-then-MAC is CCA secure, assuming Π_1 is CPA-secure and Π_2 is STRONG UF-CMA (+ proof)

(CRYPTO 2022)
Sopra presentata l'intera dimostrazione del 2022 del Thm 21 (nel 2021, Exercise).

(RECAP) From minicrypt to cryptomania



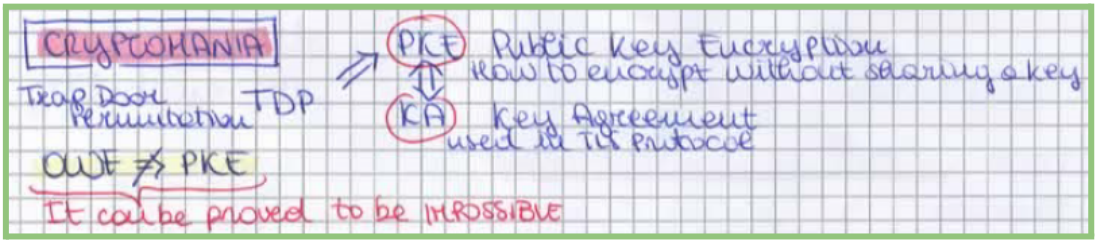
Fino adesso siamo stati nel mondo della minicrypt, dove l'argomento principale (*the main object*) erano le OWF.

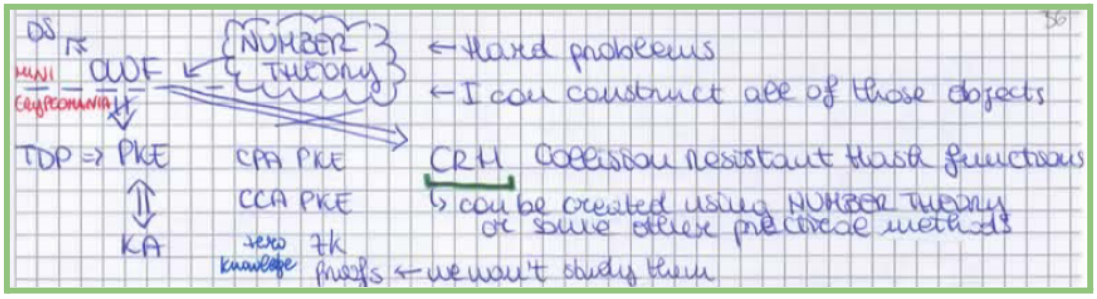


Recap minicrypt →

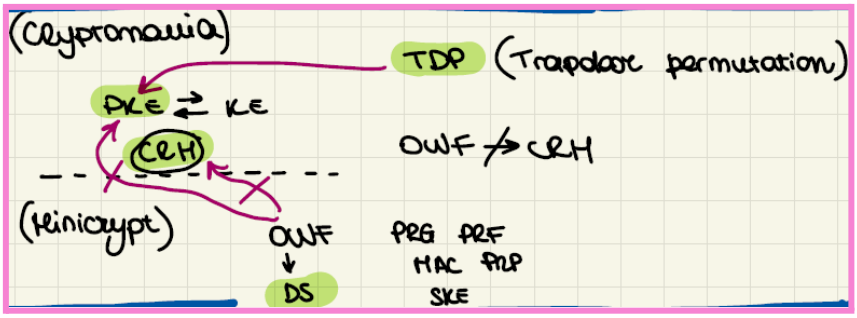
- OWF⇒PRG⇒OTS (Lecture 5, Lecture 6)
- PRFs⇒CPA secure SKE (Lecture 7)
- OWF⇒PRG⇒PRF (Lecture 7, Lecture 8)
- PRF⇒MAC (Lecture 8)
- PRF⇒PRP (Lecture 9, Lecture 10)
- PRFs⇒CCA secure SKE, CPA + Auth ⇒ CCA (Lecture 11, Lecture 12)
- OWF⇒DS
- OWF⇏CRH
- OWF⇏PKE

Adesso entreremo in un altro mondo, ovvero quello della cryptomania.





(•) We are now in cryptomania. It is the world in which there is PKE and KE (they are equivalent, PKE⇔KE). We were in minicrypt, where there are OWFs that imply everything: PRG, PRF, MAC, PRP, SKE. But we move to another world, because we cannot construct CRH from OWF (OWF⇏CRH) and it's also true that OWF doesn't imply PKE (OWF⇏PKE). It is curious that OWF implies Digital Signatures (we didn't study yet, vedi fine Lecture 19 & Lecture 20), so DS are in minicrypt (OWF⇒DS). But actually the construction of DS from OWF is inefficient, so we don't use it in real world (we can make it efficient with number theory). What is the minimal kind of assumptions for PKE? TDP. The question is can we construct PKE from something more basic, like OWF? It is proved that we cannot do it from OWF, but we can do it from trapdoor permutation. It's exactly what happens in RSA!



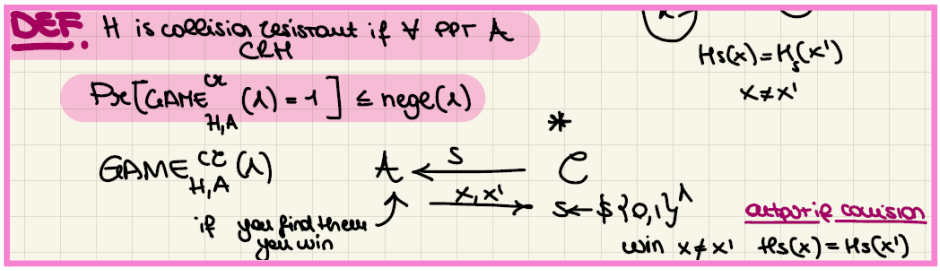
Recap cryptomania →

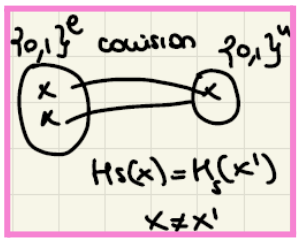
- CDH⇒DL, DL ⇒? CDH (Lecture 14)
- DDH ⇒ CDH ⇒ DL (Lecture 14)
- CDH ⇏ DDH, DL ⇏ DDH (Lecture 14)
- PKE⇔KE
- RSA⇒Factoring, Factoring⇒?RSA
- TDP⇒CPA PKE (only for 1-bit messages) (Lecture 15)
- DDH⇒CPA PKE (Lecture 16)

Collision Resistant Hashing

Definiamo ora una famiglia di funzioni hash resistenti a collisioni CRH (Collision Resistance Hash) $\mathcal{H} = \{H_s : \{0,1\}^l \to \{0,1\}^n\}_{s\in\{0,1\}^\lambda}$, con $l(\lambda) >> n(\lambda)$ e seed s pubblico. Le collisioni

esistono per il pidgeon principle (Pigeonhole principle - Wikipedia), ma devono essere difficili da trovare. $\mathcal{H}$ è una CRH family se la probabilità di collisione è negligible [Def 22 - Collision Resistant Hash Function] (in figura, il game della definizione).





Esempi di CRH nel mondo reale sono MD5, SHA-1, SHA-2, SHA-3.

- Def 22 - Collision Resistant Hash Function

(•) Tons of applications.

- $\mathcal{F}(\mathcal{H})$ is a PRF, if $\mathcal{F}$ is PRF family and $\mathcal{H}$ is CR family (universality, AU) ?

Yes, here universality suffices (Thm 14). This is domain extension for PRF ($\mathcal{F}$ is a PRF for short input size and I want to make a PRF family for long input size). Se andassi a sostituire CR, che è una secret key hash function, con una CRH, public seed, questa affermazione sarebbe comunque valida (this construction works if the hash function is secret, so if I use public hash function it's still work).
(NOTA: le sigle CR e CRH vengono utilizzate indistintamente per indicare famiglie hash resistenti a collisioni, solamente che CRH si utilizza più nell'ambito della public encryption; con CR in questo caso, si indica una famiglia hash utilizzata in un contesto di private seed, dove vale la proprietà di universality).
If I have a PRF $\mathcal{F}$, $\mathcal{F}(\mathcal{H})$ works as domain extended PRF. Of course, if I have a PRF, this is also a MAC. But I can construct Tag also not from a PRF. So consider a Tag for short input size but I have a long message, I can hash it and I obtain $Tag(\mathcal{H})$.

- $Tag(\mathcal{H})$ is MAC, if Tag is UF-CMA and $\mathcal{H}$ is CR family (universality, AU) ?

No! $Tag(\mathcal{H})$ doesn't work assuming $\mathcal{H}$ secret key hash function. But it works if $\mathcal{H}$ is public!
Exercise. $Tag(\mathcal{H})$ is MAC, if Tag is UF-CMA and $\mathcal{H}$ is CRH.
Tag is not like PRF because to be UF-CMA you don't need to be RAND. NOTA: being unforgeable is implied by being random/pseudorandom (I cannot predict it, so I cannot forge), but I could have a Tag unforgeable that is not random/pseudorandom.

SUBTLETY. There is no seed in the real world! The seed makes the definition possible. We will see the main principles behind real world constructions. No function is secure in the notion above if it does not have a seed so the real world construction cannot be proved to be secure in that notion. We can only try to break them but until then we consider them "secure". Why is no function secure in the notion above? Because whatever the function is, exists an adversary that already knows a collision, but in practice we don't know how to find this collision so this does not mean that the function is not secure.
Why the seed? (•) Can we have a single function and not a family? No, we need seed (so the family), because there always exists a bad adversary which knows the collision (the adversary knows the function, that is public). So if I fix a single function, by definition there are collisions, but also exists a bad adversary which already knows the collision: it's like a machine that inside has the code to find the collision. In practice it doesn't really matter because we don't know how to write the code of this machine. We can't do the same

reasoning for a family because in the definition we first fix the adversary and then we take the seed. So if you fixed $\mathcal{H}$ family of function, it exists always an adversary $A_{x,x'}$ that outputs collision x, x'. The seed in this sense is the description of the hash function, because it fixes the hash function.

Lecture 13

[1]: 4.2, 4.4, B, C [2]: 5.2, 5.5, 5.6.2, 6.3.1, 8.1, B [7]: 3.6, 4

- ➢ The Merkle-Damgaard and Merkle tree constructions.
- ➢ Constructing secure compression functions: The Davies-Mayer construction and its security in the ideal cipher model.
- ➢ Brush-up on number theory: Modular arithmetic, Euclidean algorithm, prime numbers.
- ➢ Integers factorization.

(•) Recipe (MD-5, SHA-1, SHA-2, SHA-3 all follow this recipe):

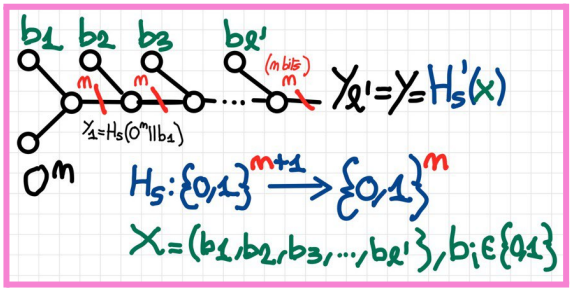
a. Build CRH for FIL (2). Set finest "compression function" with fixed compression ($l \to n$) (or fixed input length) …
   fixed input length → fixed input length, e.g. $n + 1$ bits → n bits o $2n$ bits → n bits;
b. Use (a) to be CRH for VIL (any compression) (1, 1A Merkle-Damgaard and 1B Merkle Tree). Bootstrap in to $\{0,1\}^*$ …
   $\{0,1\}^*$ → fixed input length.

Otteniamo collision resistance in due step:
1. Domain extension; 2. Fixed amount of compression.
Iniziamo con il punto 1: how do you get hash functions for VIL if you have compression functions for FIL? Abbiamo due tecniche: 1A MERKLE-DAMGAARD e 1B MERKLE-TREE.

The Merkle-Damgaard and Merkle tree constructions

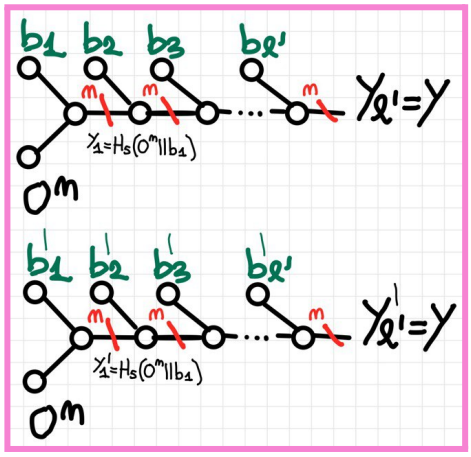
- 1A MERKLE-DAMGAARD

La metodologia di Merkle-Damgard ci consente di costruire una funzione hash resistente alle collisioni a partire da una qualsiasi funzione di compressione per stringhe a lunghezza fissa (resistente alle collisioni) ([1]: 4.2 (La costruzione di Merkle-Damgård)). H_s (nella costruzione a lato) è una collision resistant hash function e comprime una quantità fissa (one bit of compression). Al primo step, come mostrato in figura, y_1 è l'hash risultato della concatenazione di n zeri con il primo bit. Ad ogni step quindi il bit b_i e l'y_{i-1} vanno a costituire l'hash y_i.



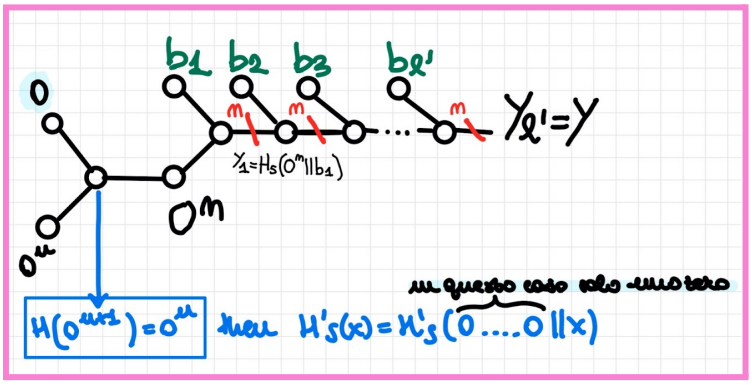
Definiamo un teorema per cui se $\mathcal{H} = \{H_s : \{0,1\}^{n+1} \to \{0,1\}^n\}_{s\in\{0,1\}^\lambda}$ è CRH, allora $\mathcal{H}' = \{H_s' : \{0,1\}^{l'} \to \{0,1\}^n\}_{s\in\{0,1\}^\lambda}$ (Merkle-Damgard construction) è CRH per ogni fixed (arbitrary length but fixed) $l' = poly(\lambda)$ [Thm 22 - The MD construction gives a CRH $\mathcal{H}'$ from $l'(\lambda) \to n(\lambda)$ assuming $\mathcal{H}$ is CRH from $n + 1 \to n$]. Per la dimostrazione assumiamo per

assurdo che un avversario PPT A, dato s, dia in output x, x', con $x = (b_1, \dots, b_{l'}) \neq (b_1', \dots, b_{l'}') = x'$, tali che si abbia una collisione, ovvero quindi $H'_s(x) = H'_s(x')$ (MD hash). Sia quindi y il risultato della funzione hash H'_s sia per x che per x'. Se andiamo a ritroso (running backwards), ci deve essere un i allora tale che $(b_i, y_{i-1}) \neq (b_i', y_{i-1}')$ e $H_s(b_i, y_{i-1}) = H_s(b_i', y_{i-1}')$ (NOTA: questo i sarà l'indice più grande affinché questa proposizione sia rispettata). Questo contraddice l'ipotesi di collision resistance di H_s. Abbiamo quindi trasformato una collisione per H' in una collisione per H (given any collision for H', I can just recompute the steps of MD and then going backwards I can find a pair such that it's a collision for H). Questa osservazione implica una semplice reduction (it transforms a collision of H' in a collision for H).



- Thm 22 - The MD construction gives a CRH $\mathcal{H}'$ from $l'(\lambda) \to n(\lambda)$ assuming $\mathcal{H}$ is CRH from $n + 1 \to n$ (+ proof)

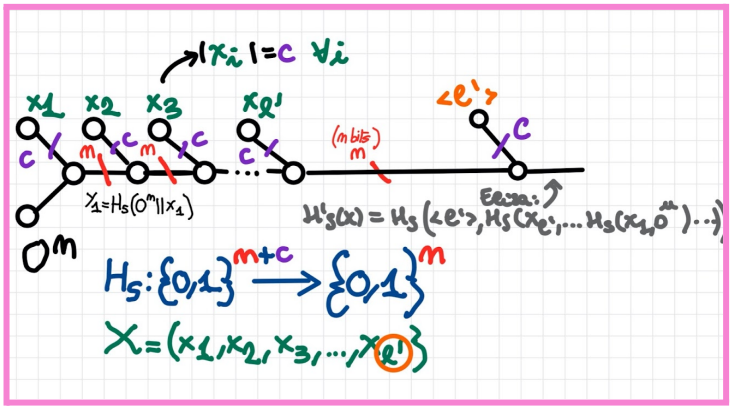
Questa costruzione funziona solo per FIL, mentre è insicura per VIL (it means that exists a *BAD* H_s which is a secure compression function from $n + 1 \to n$ but for which this structure is not secure for VIL). Why? Assume $H_s(0^{n+1}) = 0^n$ for each s (this function does not contradict collision resistance), so if $x' = 0||x$ (length: $n + 1$) and x (length: n) then $H_s'(x) = H_s'(0||x)$ (same output, $0 = 0$). Problem: x is a suffix of x'!



How to fix it? Assicuriamoci che nessun input ammissibile possa essere un suffisso di un altro input ammissibile (SUFFIX-FREE ENCODING, no legal input is a suffix of another legal input). The actual MD use suffix-free encoding of x. (Come possiamo vedere dalla figura) We're append to x an encoding $< l' >$ (binary representation of $l' \in \mathbb{N}$ with c bits) of its length.

(•) Nota. Binary representation of l' must be written in c bits, so the only constraint is that I can only do VIL as long as the max length of the input is long in binary c bits (but c is very big, so it doesn't really matter!). This construction works when the length of x is a multiple of c; if it isn't multiple, you can pad the message.

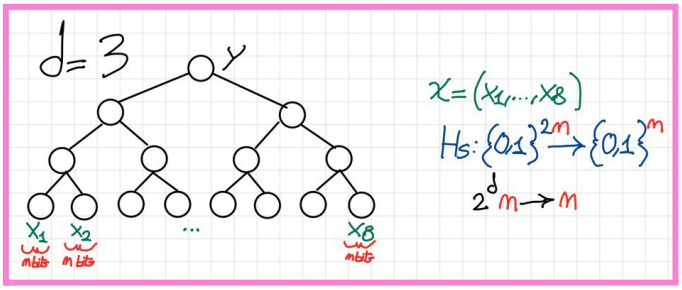
Dimostriamo quindi che questa costruzione (strengthen MD) è CR per VIL input [Thm 23 - The strengthen MD is CR for VIL inputs]. Assumiamo per assurdo che un avversario PPT A, dato s, dia in output x, x', con $x = (x_1, \dots, x_l) \neq (x_1', \dots, x_{l''}') = x'$, tali che si abbia una collisione, ovvero quindi $H'_s(x) = H'_s(x')$; abbiamo quindi due casi:



- se $l' = l''$ → ci ritroviamo nella situazione di prima (FIL, same proof as before);
- se $l' \neq l''$ → allora prendendo in considerazione $H'_s(x)$ e $H'_s(x')$ notiamo che si avrà $H_s(y_{l'}, < l' >) = H_s(y_{l''}, < l'' >)$ all'ultimo step di MD proprio perché abbiamo supposto che $H'_s(x) = H'_s(x')$ (reduction to collision of H), ma la probabilità di collisione in H è negligible e quindi abbiamo provato che H' è sicura.

- Thm 23 - The strengthen MD is CR for VIL inputs (+ proof)

- 1B MERKLE-TREE



Questa costruzione ci permette di fare l'hash di una stringa molto lunga in una molto corta. Proof that this works is easy. H_s (in figura) è una collision resistant hash function.

(•) Now we know how to bootstrap!

Constructing secure compression functions

2 BUILD COMPRESSION FUNCTION

- Practice

We use AES and the Davies-Meyer construction. Let's define the (block) cipher (AES function) as $E_{x_1}(x_2)$. Then, $H_E(x_1, x_2) = E_{x_1}(x_2) \oplus x_2$ ($2n \to n$ or even $n + \lambda \to n$). To be CR, it should be hard to find $(x_1, x_2) \neq (x_1', x_2')\ s.t.\ E_{x_1}(x_2) \oplus x_2 = E_{x_1'}(x_2') \oplus x_2'$.

There are some problems: it's suspicious (block cipher is a PRP, but if an OWF gives PRP, then we should get CRH from OWF but this is impossible) and this is insecure for concrete PRP. This second problem is due to the fact that, given AES, the key is part of the input and thus the key can be, in a certain way, chosen by the adversary. We can break this with a strange cipher: if we take E such that $E_{0^n}(0^n) = 0^n$, $E_{1^n}(1^n) = 1^n$ this will lead to a collision because we'll have that $E_{0^n}(0^n) \oplus 0^n = 0^n$ and the same is for $E_{1^n}(1^n) \oplus 1^n = 0^n$.

To have a truly random permutation (random truth table) for every key we should pick the Ideal Cipher Model (ICM) which doesn't exist in real life.
Observation. AES is CR? No one has proven this.

- Theory

(•) There are many examples. An easy one (vedi Lecture 15, costruzione CRH da DL):
$params = (\mathbb{G}, g, q, y)$ (e.g. $p = 2q + 1$, $\mathbb{G} = \mathbb{QR}_p$),

$H_{p,q,y}(x_1, x_2) = g^{x_1} y^{x_2}$ $(2\lambda \to \lambda)$, $y \leftarrow \$\mathbb{G}$.

Number theory

(•) We started with Alice and Bob that want to do secure communication by sharing a short key: we can do in the computational world by just assuming OWF. However this is not very practical. With number theory we can move in cryptomania! Here to construct PRG, PRF becomes trivial. In this world you can also do:

- KE (Key Exchange) → Alice and Bob now share a key, without actually meet in a secret place (real world);
- PKE (Public Key Encryption);
- CRH (Collision Resistant hash function) (Lecture 12).

Another thing to say is that while in minicrypt we have alternatives, number theory is the only way to do PKE if you want to do it (no alternatives).
Vedremo quindi costruzioni crittografiche radicate nella number theory (teoria dei numeri). La number theory è una buona sorgente per hard problems e ci permette di costruire direttamente molte cose.

[1]: B (Teoria dei numeri)
La teoria dei numeri riguarda lo studio delle proprietà dei numeri interi. La formulazione di molti dei problemi tipici della materia può essere facilmente compresa anche da un non-matematico; tuttavia la soluzione degli stessi problemi è spesso complessa, e richiede tecniche non banali. La teoria dei numeri nasce nell'antica Grecia, con lo studio dei criteri di divisibilità degli interi (da parte di Euclide ad esempio). Più avanti (nel XVII secolo) fu Pierre de Fermat ad iniziare uno studio più sistematico. Il matematico Francese, in particolare, enunciò numerose congetture, senza fornire per esse una dimostrazione. La più famosa di queste è senz'altro il famoso "ultimo teorema" (Fermat Last theorem), che afferma che non esistono soluzioni intere all'equazione $x^n + y^n = z^n$ per $n > 2$. La soluzione dell'enigma di Fermat ha visto impegnate, senza fortuna, intere generazioni di matematici. La congettura è stata dimostrata da Andrew Wiles, nel 1994. I contributi successivi, ad opera di Eulero, Lagrange e Gauss, hanno quindi dato inizio alla teoria dei numeri moderna.

Modular arithmetic

(•) Let's introduce the main set up of the number theory: modular arithmetic ([1]: B.2 (L'aritmetica dell'orologio)). We'll study two main structures ($+$ and $\cdot$ of $\mathbb{Z}_n$).

Consideriamo $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$ (o possiamo dire anche integers $mod\, n$). Any number $mod\, n$ is the remainder of that number when is divided by n.
Example. $n = 4$; $2\, mod\, 4 = 2$, $4\, mod\, 4 = 0$, $5\, mod\, 4 = 1$

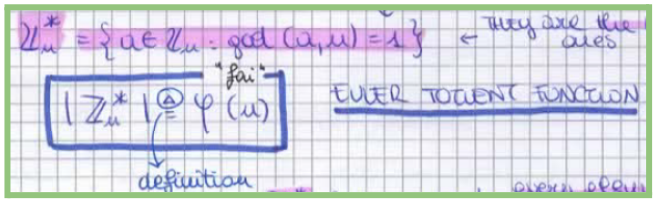
$(\mathbb{Z}_n, +)$ è un gruppo (group):

- (i) identity: $\exists 0 \in \mathbb{Z}_n \; s.t. \; a + 0 = a \; mod \; n \; \forall a \in \mathbb{Z}_n$
- (ii) add: $a + b = b + a \in \mathbb{Z}_n$
- (iii) inverse: $\forall a, \; \exists \; (-a) \in \mathbb{Z}_n \; s.t. \; a + (-a) = 0$

$(\mathbb{Z}_n, \cdot)$ è un gruppo? No, non è un gruppo in generale (no inverse). Per provare che $(\mathbb{Z}_n, \cdot)$ non è un gruppo, definiamo un lemma [Lemma] in cui affermiamo che se il massimo comune divisore tra a e n è diverso da 1 (> 1), allora a non è invertibile. Proviamo il lemma assumendo per assurdo che a sia invertibile, quindi che esiste un $b \in \mathbb{Z}_n$ tale che $ab \equiv 1 \; mod \; n$. Allora: $ab = 1 + qn \rightarrow ab - qn = 1$, per qualche intero q. Evidentemente $gcd(a, n)$ divide $ab - qn$ e quindi deve anche dividere 1, da cui $gcd(a, n) = 1$ contro l'ipotesi ([1] Lemma B.9 (Elementi invertibili in $\mathbb{Z}_n$)).

- Lemma - If $gcd(a, n) \neq 1$, then a not invertible (+ proof)

Si definisce con $\mathbb{Z}_n^*$ l'insieme dei residui invertibili modulo n (subgroup of $\mathbb{Z}_n$ with the invertible ones). Applicando il lemma, sappiamo che: $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n : gcd(a, n) = 1\}$. Notare che la cardinalità di $\mathbb{Z}_n^*$ è data dal numero di elementi minori di n e coprimi con esso; si definisce la funzione toziente di Eulero $\varphi(n)$ come il numero di elementi minori di n e coprimi con esso ([1]: Definizione B.5 (Funzione toziente di Eulero)). Se ad esempio scegliessimo $n = p$ con p uguale ad un numero primo, allora $\mathbb{Z}_p^* = \{1, 2, ..., p - 1\}$, ovvero ogni elemento sarebbe invertibile ($gcd(a, x) = 1, \; \forall x \in \mathbb{Z}_p^*$) ad eccezione di 0 e avremmo $\varphi(p) = p - 1$. For us: we will work with $\mathbb{Z}_n^*$, $\mathbb{Z}_p^*$ and more with huge n or p ($|n| = 2048$ bits).



Euclidean algorithm

Abbiamo bisogno di una implementazione efficiente delle operazioni $mod \; n$. Moltiplicazione e addizione in $\mathbb{Z}_n$ sono efficienti. L'inverso di un elemento non esiste sempre, ma solo se $gcd = 1$ e può essere calcolato con l'extended Euclidean algorithm [Lemma]. Definiamo quindi il lemma per cui dati due interi a e b tali che $a \geq b > 0$, allora $gcd(a, b) = gcd(b, a \; mod \; b)$. Per dimostrare il lemma, è sufficiente mostrare che i divisori comuni ad a e b sono gli stessi comuni a b ed $a \; mod \; b$. Scriviamo $a = qb + a \; mod \; b$, dove q è il quoziente e $a \; mod \; b$ è il resto della divisione tra a e b ([1]: Lemma B.1 (Quoziente e resto)). Pertanto, un divisore comune ad a e b divide anche $a - qb = a \; mod \; b$. D'altra parte, un divisore comune di b ed $a \; mod \; b$ divide anche $a = qb + a \; mod \; b$; quindi l'asserto ([1]: Lemma B.2 (Algoritmo di Euclide)).

- Lemma - Let $a, b \; s.t. \; a \geq b > 0$ then $gcd(a, b) = gcd(b, a \; mod \; b)$ (+ proof)

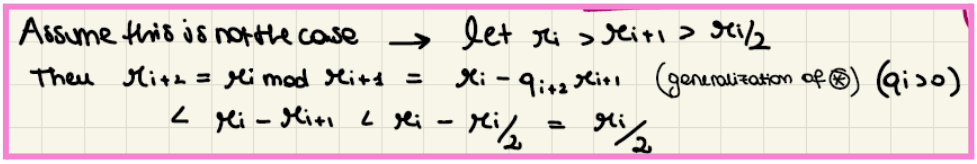
Definiamo quindi il teorema per cui dati due interi a e b possiamo computare $gcd(a, b)$ in tempo polinomiale ([1]: Corollario B.5 (Complessità dell'algoritmo di Euclide)) ed inoltre possiamo trovare due interi u e v tali che $gcd(a, b) = ua + bv$ ([1]: Teorema B.6 (Identità di Bézout)) [Thm

24 - Given a, b we can compute $gcd(a,b)$ in polynomial time ($O(log^3\lambda)$) and also we can find $u, v \ s.t. \ gcd(a,b) = ua + bv$]. Preso $a \epsilon \mathbb{Z}_n$ tale che $gcd(a, n) = 1$, sono presenti u, v tali che $ua + vn = gcd(a,n) = 1$ (vn, ovvero un multiplo di n) e quindi $gcd(a,n) = ua \bmod n \rightarrow$ abbiamo dunque trovato l'inverso di a che è n! (•) In other words, if $b = n$ then $gcd(a,n) = 1 = ua + multiple\ of\ n = ua \bmod n$ and so $a^{-1} = n$.
Dimostriamo quindi che $gcd(a,b)$ è eseguibile in polynomial time, utilizzando (Lemma - Let $a, b \ s.t. \ a \geq b > 0$ then $gcd(a,b) = gcd(b, a \bmod b)$) in maniera ricorsiva.
Quindi applichiamo l'algoritmo:
$a = b * q1 + r1$, $0 \leq r1 \leq b$ e $gcd(a,b) = gcd(b, a \bmod b) = gcd(b, r1) \rightarrow$
ora ricorsivamente $b = r1 * q2 + r2$ ➔ $gcd(b, r1) = gcd(r1, b \bmod r1) = gcd(r1, r2)$,
fintanto che per una qualche t, $r_{t+1} = 0$ in modo tale che $gcd(a,b) = r_t$ (gcd is the last non-zero remainder).
Mostriamo quindi che l'algoritmo termina in $poly(\lambda)$ step.
(•) Clearly, we know that $r_{i+1} < r_i$, but it doesn't suffice to conclude that the algorithm runs in polynomial time, because for instance if you take $a = 2^\lambda$ bit and it decreased by 1 then you take 2^λ steps (exponential number of steps). Dobbiamo quindi trovare un caso in cui gli step siano in numero polinomiale.

Affermiamo e dimostriamo quindi (CLAIM) che $r_{i+2} \leq \frac{r_i}{2}$, $\forall i = 0, \dots, t-2$.

(•) The claim means that every two steps, the remainder halves, i.e. we lose a bit ($\#bit\ of\ n = \lfloor log_2 n \rfloor + 1$ ➔ $\#bit\ of\ \frac{n}{2} = \lfloor log_2 \frac{n}{2} \rfloor + 1 = \lfloor log_2 n \rfloor$). So if a and b are λ bit integers, we do approximately $2(\lambda - 1)$ steps. At every step, we do a division that is also $poly(\lambda)$ time.

Adesso se abbiamo $r_{i+1} \leq \frac{r_i}{2}$, la dimostrazione è fatta; se assumiamo invece per assurdo che $r_{i+1} > \frac{r_i}{2}$ …



- Thm 24 - Given a, b we can compute $gcd(a,b)$ in polynomial time ($O(log^3\lambda)$) and also we can find $u, v \ s.t. \ gcd(a,b) = ua + bv$ (+ proof + CLAIM with proof)

Example. $a = 14$, $b = 10$, $gcd(a,b)$?
$a = b * q1 + r1 \quad \Rightarrow 14 = 1 * 10 + 4 \quad \rightarrow gcd(a,b) = gcd(b, a \bmod b) = gcd(b, r1)$
$b = r1 * q2 + r2 \quad \Rightarrow 10 = 2 * 4 + 2 \quad \rightarrow gcd(b, r1) = gcd(r1, b \bmod r1) = gcd(r1, r2)$
$r1 = r2 * q3 + r3 \quad \Rightarrow 4 = 2 * 2 + 0 \quad \rightarrow r3 = 0, STOP!\ gcd(4,2) = gcd(14,10) = 2$
How to find u e v? → Reverse the steps! (poly time)
$2 = 10 - 2 * 4 = 10 - 2(14 - 1 * 10) = 3 * 10 + (-2) * 14$ ➡ $u = 3, v = (-2)$

RECAP. Until now …
- $(\mathbb{Z}_n, +)$ is a group
- $(\mathbb{Z}_n^*, \cdot)$ is a group

- In polynomial time, we can do:
  - Addition - Multiplication - Inverse (if exists) ($\rightarrow gcd(a, n) = 1$)

With Extended Euclidean Algorithm, we can find and compute the inverse in poly time.
We can also demonstrate that the exponentiation can be done in poly time.

Modular exponentiation

A cosa è uguale $a^b \bmod n$? Scriviamo b in binario ($b = (b_l, \ldots, b_0)$) e otteniamo …

$$a^b \bmod n = a^{\sum_{i=0}^{l} b_i * 2^i} \bmod n$$

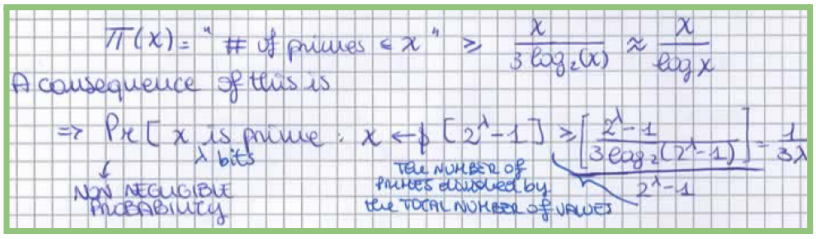
$$= \Pi_{i=0}^{l} a^{b_i * 2^i} \bmod n \quad \text{(when } b_i = 0 \rightarrow a^0 = 1\text{, quindi non contribuisce al prodotto)}$$

$$= a^{b_0} * (a^2)^{b_1} * (a^4)^{b_2} * \ldots (a^{2^l})^{b_l} \bmod n$$

(Square [(•) l times] and multiply)

(•) In questo modo abbiamo provato che preso $a^b \bmod n$ ci vuole polynomial time per fare l'esponenziale $\bmod n$.

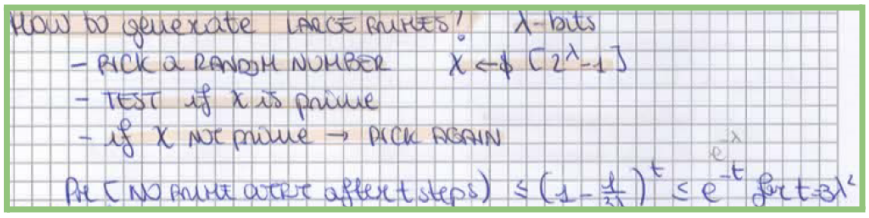
Prime numbers



Esistono infiniti numeri primi [Thm 25 - Prime Number Theorem]. Preso un numero randomico x composto da λ bit, la probabilità che questo sia primo non è negligible: generating random number, we have a good chance that it's a prime! Ma come capiamo se x è primo? → Primality testing (Muller-Rabin 80', Aks '02) [Thm 26 - We can test if a λ-bit integer is prime in $poly(\lambda)$ time]. (•) This one is a probabilistic algo.

- Thm 25 - Prime Number Theorem
- Thm 26 - We can test if a λ-bit integer is prime in $poly(\lambda)$ time

How do we generate large primes? La probabilità di non avere in output dopo t step (generando numeri random) un numero primo è negligible.



[telegram]: Theorem 13.100 (Number theory)
There exist many algorithms that solve the problem of primality testing, with time complexity polynomial in the length of their numerical representation; of most relevance are those of Miller-Rabin, which is probabilistic, but consistently used in practice, and the completely deterministic Agrawal-Kayal-Saxena (aks), which has a much greater polynomial rank, and has been deemed impractical for most uses.

Factorization

Procediamo con il nostro primo esempio di OWF. Sia $n = p * q$ con p, q di dimensione λ bit, allora $n = f(p,q) = p * q$ è una OWF ($p * q$ può essere computata efficientemente e n dati p, q non può essere computata - NP problem). Brute force (try all primes, $\leq \sqrt{n}$) e altri tipi di attacco sono possibili, ma non sono efficaci (still very far from polynomial time because exponential).
ASSUMPTION/CONJECTURE (→ everyone believes in it, but it's a hard problem). Integer factorisation of the product of two λ bit primes is a One Way Function (OWF) ([7]: 4 (Number Theory)).
(•) Osservazione. RSA (one of the first PKE) is based on integer assumption

Lecture 14

[1]: B, C [2]: 8.3, B [7]: 4.1, 4.2

- ➢ Lagrange's theorem.
- ➢ Cyclic groups.
- ➢ The Discrete Logarithm (DL) problem.
- ➢ Diffie-Hellman key exchange.
- ➢ Computational Diffie-Hellman (CDH) and Decisional Diffie-Hellman (DDH) assumptions.
- ➢ Hardness of DDH.

(CRYPTO 2022)
Nel 2022, dimostrazione Thm 27 non fatta. Corollario con dimostrazione sì.

Lagrange's theorem

[Thm 27 - Lagrange's theorem] Sia $\mathbb{H}$ un sottogruppo di un gruppo finito $\mathbb{G}$. Allora $\#\mathbb{H}$ divide $\#\mathbb{G}$ (the cardinality of $\mathbb{H}$ divides the cardinality of $\mathbb{G}$ → $|\mathbb{H}| \,|\, |\mathbb{G}|$, i.e. $|\mathbb{G}|$ è un multiplo di $|\mathbb{H}|$) ([1]: Teorema B.8 (Teorema di Lagrange)).
Come conseguenza del teorema, abbiamo un corollario [Corollario] che afferma che per ogni $a \epsilon \mathbb{Z}_n^*$ abbiamo che:

1. $a^{\varphi(n)} \equiv 1 \, mod \, n$ [Euler's theorem] ($n = p, a^{p-1} \equiv 1 \, mod \, p$, Fermat's Little theorem)
2. $a^b = a^{b \, mod \, \varphi(n)} \, mod \, n$

(•) Proviamo quindi il corollario.
(*1*) ($\mathbb{Z}_n^*$, *) is a group with $\varphi(n)$ elements. Consider $a \epsilon \mathbb{Z}_n^*$ and the subgroup of the powers of a: $< a >= \{a^0, a^1, .., a^{d-1}\}$ ($a^d = 1$, where d is the order of this subgroup). By Lagrange, $| < a > | \,|\, |\mathbb{Z}_n^*|$ (Thm 27, $|\mathbb{H}| \,|\, |\mathbb{G}|$), so $d | \varphi(n)$, i.e. $d * k = \varphi(n)$ for some k. Then $a^{\varphi(n)} = (a^d)^k = 1^k = 1 \, mod \, n$.
(*2*) Also, $a^b = a^{q*\varphi(n)+b \, mod \varphi(n)} = a^{b mod \varphi(n)} mod \, n$, because $(a^{\varphi(n)})^q = 1$.

- Thm 27 - Lagrange's theorem + Corollario with Euler's theorem (+ proof)

Cyclic group

For us: we will mainly work in $\mathbb{Z}_n^*$ with $n = p * q$ e $\mathbb{Z}_p^*$ with p prime.

(•) Adesso ci spostiamo dall'analisi di $\mathbb{Z}_n^*$ a quella di $\mathbb{Z}_p^*$ poiché vogliamo arrivare ad una assunzione che vale solo per $\mathbb{Z}_p^*$; analizziamo da più vicino quindi $\mathbb{Z}_p^* \rightarrow$

$(\mathbb{Z}_p^*, +, *)$ is a field because $\forall a \epsilon \mathbb{Z}_p^*, gcd(a, p) = 1$.

Example. $gcd(5, 11) = 1$; $1 = 11 - 2 * 5$. Quindi l'inverso di 5 (5^{-1}) coincide con (-2) che coincide con $9\ mod\ 11$. Infatti $9 * 5 = 45 = 1\ mod\ 11$.

$(\mathbb{Z}_p^*, *)$ è un gruppo ciclico (cyclic group). Quindi esiste un g un generatore (generator) appartenente a $\mathbb{Z}_p^*$ tale che $\mathbb{G} = \{g^0, g^1, .., g^{p-2}\}$ (con $g^0 = g^{p-1}$).

Example.

- 3 is a generator of $\mathbb{Z}_7^* \rightarrow g = 3$, $\{3^0, 3^1, .., 3^{7-2}\} = \{1, 3, 2, 6, 4, 5\}$
- 2 is NOT a generator of $\mathbb{Z}_7^* \rightarrow 2^3 \equiv 1\ mod\ 7$, quindi non genera tutti i numeri da 1 a 6.

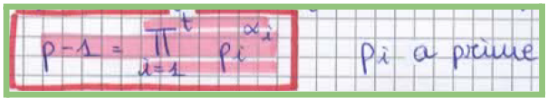
(•) Assunzioni fatte finora su $\mathbb{Z}_p^*$: tutti gli elementi di $\mathbb{Z}_p^*$ sono invertibili e $\mathbb{Z}_p^*$ è un gruppo ciclico (that it means that exists a generator that generates all group).

(CRYPTO 2022)
Nel 2022, factorization in maniera formale (di seguito) non fatta.

FACTORIZATION

Come generiamo $\mathbb{Z}_p^*$ lungo un generatore g? È possibile data la fattorizzazione di $p - 1$. La fattorizzazione viene eseguita in tempo polinomiale anche se il numero è grande. Dato che $p_i \geq 2$ allora $2^t \leq p \leq 2^{\lambda+1}$. Quindi $t \leq \lambda$. Dal teorema di Lagrange, l'ordine di $y \epsilon \mathbb{Z}_p^*$ deve dividere (e deve essere minore di) $p - 1$ se e solo se (IFF) y non è un generatore:

$y\ not\ a\ generator \leftrightarrow \exists i,\ 1 \leq i \leq t \quad y^{\frac{p-1}{p_i}} \equiv 1\ mod\ p.$

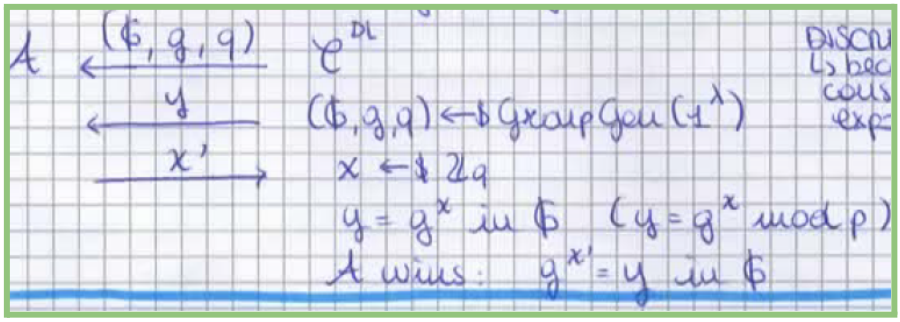


Example. $p = 7$, $p - 1 = 6 = 3 * 2$
y is a generator IFF $y^2\ mod\ 7 \neq 1$ and $y^3\ mod\ 7 \neq 1$
3 is a generator. 2 is NOT a generator.

The Discrete Logarithm (DL) problem

Procediamo con il nostro secondo esempio di OWF. [(•) La definiamo perchè tra poco ci sposteremo da $\mathbb{Z}_p^*$.]

Definiamo quindi DL (discrete log) tramite $(\mathbb{G}, g, q) \leftarrow \$GroupGen(1^\lambda)$, con $\mathbb{G}$ cyclic group, g generator, q order ($q = |\mathbb{G}| = p - 1$, $g^q = 1\ in\ \mathbb{G}$) e $GroupGen(1^\lambda)$ l'algoritmo.



[1]: C.3 (Il logaritmo discreto)

Logaritmi in $\mathbb{Z}_p^*$. Sia p un primo. Come abbiamo visto nell'Appendice B.2 il gruppo $\mathbb{Z}_p^*$ è ciclico e si chiama generatore un elemento g che ha ordine $p-1$ in $\mathbb{Z}_p^*$. Dato un tale generatore, ogni elemento $y \in \mathbb{Z}_p^*$ può essere espresso come $y = g^x$ per un qualche intero x.

Definizione C.2 (Logaritmo discreto in $\mathbb{Z}_p^*$). Con la notazione sopra introdotta, diremo che x è il logaritmo discreto di y rispetto alla base g e scriveremo $x = log_g y$. Osserviamo che per il piccolo teorema di Fermat (cf. Teorema B.11) possiamo sempre aggiungere multipli di $p-1$ all'esponente, ovvero: $y = g^x \equiv g^{x+k(p-1)} \; mod\, p$. Segue che l'esponente $x = log_g y$ è sempre un elemento di $\mathbb{Z}_{p-1}$.

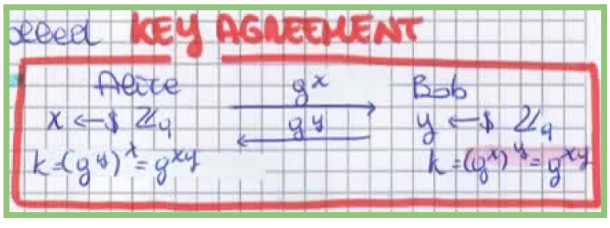
ASSUMPTION/CONJECTURE. DL (discrete log) defines a OWF ($f_{DL}(x) = y = g^x$ in $\mathbb{G}$ is a OWF).

(•) For every x, $f_{DL}(x) = y = g^x$ in $\mathbb{G}$. If $\mathbb{G} = \mathbb{Z}_p^*$ then $y = g^x mod\, p$. Now the task of Eve is to compute x, the discrete log. Because the group is cyclic there exists a x for every y. We brute force $x = 0, 1, 2, \ldots, q-1$ and this takes exponential time.

Discrete log and integer factorization are both examples of OWF and there exist many algorithms to solve them which take subexponential time. Discrete log is considered "better" than integer factorization.

Diffie-Hellman key exchange

This is like the beginning of cryptomania: we'll start from the problem of key exchange. With this protocol we begin Public Key Encryption. Alice and Bob exchange a key over a public channel. How to do this? There exist a public group $(\mathbb{G}, g, q) \leftarrow \$GroupGen(1^\lambda)$ where for instance if $\mathbb{G} = \mathbb{Z}_p^*$ then the operation is $mod\, p$ and the exponents are between 0 and $q = p - 1$. Now, what happens?



- Alice and Bob pick two random exponents (respectively x and y) and they exchange g^x and g^y.
- Alice computes the key k by receiving g^y ($k = (g^y)^x$) and Bob viceversa ($k = (g^x)^y$).

(•) What about Eve? Eve knows the parameters (the parameters are public, because we've a public channel) and the two exchanged values: $Eve(params, g^x, g^y)$. Eve is passive:

(1) Eve can't compute the key k (maybe can compute a part of it);
(2) Eve thinks that k is random over $\mathbb{G}$.

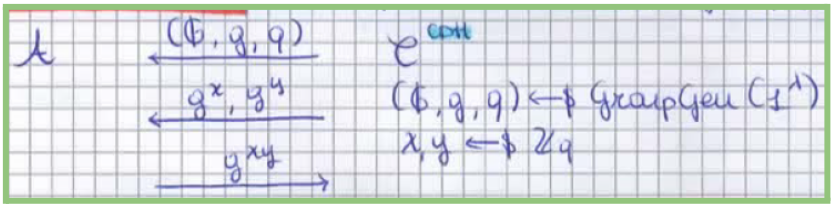
Is this secure? A protocol (in which the parties share the key) is secure when (1) it's hard to compute the key for an adversary or (2) the key looks random (better notion, stronger).
(•) Options (1) is not very strong, because maybe Eve cannot compute the key, but can compute the first bit of the key or obtain info about it. For instance, it's dangerous if Eve knows half of the key. So better the option (2): Eve here has no proof of what the key is, i.e. from her point of view the key is an uniform random element of the group. Both options require DL (if DL problem would be easy, this protocol is not secure, because Eve can compute x from g^x and y from g^y).

CDH and DDH

We have two important ASSUMPTIONS (under these, the protocol is secure). $\forall PPT\ Eve\ldots$

1. CDH - Computational DH

(•) … $Pr[Eve(params, g^x, g^y) = g^{xy}] = negl(\lambda)$ (the probability that Eve can compute the key is negligible, (1))

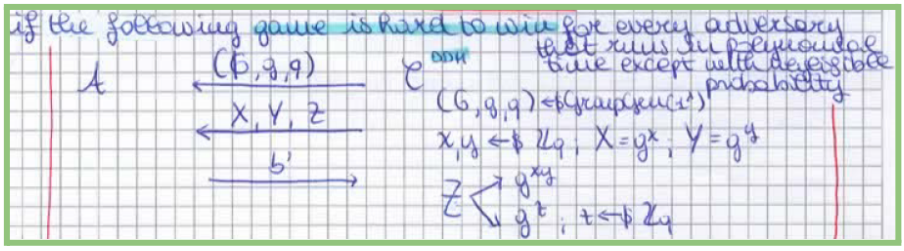


CDH⇒DL, because CDH holds wrt (with respect to) $GroupGen$.

(•) (Dimostrazione per assurdo) Per CDH, abbiamo che $Pr[Eve(params, g^x, g^y) = g^{xy}] = negl(\lambda)$. Assumiamo per assurdo che DL non sia hard, ovvero che possiamo calcolare x e y conoscendo g^x e g^y. Ma se questo è vero possiamo anche calcolare g^{xy} poiché conosciamo x e y. Quindi arriviamo ad un assurdo, perché la probabilità che si riesca a calcolare g^{xy} è negligibile per CDH, quindi se DL non fosse hard, staremo negando $Pr[Eve(params, g^x, g^y) = g^{xy}] = negl(\lambda)$.
The other side we don't know (DL⇒?CDH), so we don't know if breaking CDH is equivalent to breaking DL.

2. DDH - Decisional DH

… $(params, g^x, g^y, g^{xy}) \approx_c (params, g^x, g^y, g^z)$ with x, y, z random on $\mathbb{Z}_q$ (stronger notion because not only it is hard to compute the key but actually g^{xy} is random, (2))



DDH⇒CDH⇒DL, because DDH holds if for an adversary A that runs in poly time is hard to win a game in which he needs to recognize between g^{xy} and g^{z}.

(•) (Dimostrazione per assurdo) Per DDH, abbiamo che $(params, g^x, g^y, g^{xy}) \approx_c (params, g^x, g^y, g^z)$. Assumiamo che CDH non sia hard: se CDH non è hard possiamo calcolare g^{xy} con probabilità maggiore di negligible. Questo significa che g^{xy} è computabile e di conseguenza, se questo è vero, è possibile distinguerlo da g^z, quindi $(params, g^x, g^y, g^{xy}) \approx_c (params, g^x, g^y, g^z)$ non è più vera. Quindi assumendo CDH non hard, stiamo negando DDH e arriviamo all'assurdo.

Fact: DDH does not hold in $\mathbb{G}= \mathbb{Z}_p^*$. In $\mathbb{Z}_p^*$, we believe that CDH and DL are hard, however DDH is not hard. In fact CDH does not imply DDH (CDH⇏DDH) and it follows that DL does not imply DDH (DL⇏DDH).

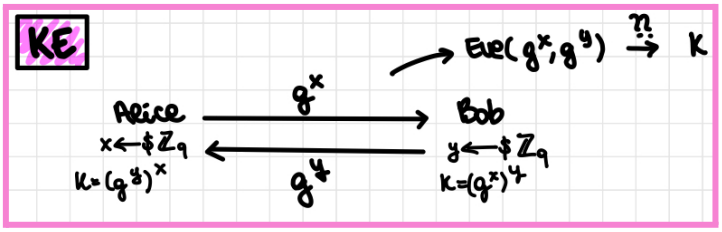
(•) RECAP

Number Theoretical assumptions most used:

- Factoring, $n = pq$ is a OWF with p, q prime numbers of λ bit (NOTA: λ → security parameter, usually the length of the key [$\lambda = 1024\ bits$])

→ In factoring assumption, n is very large: new schemes are based on other assumptions (DL more efficient).

- DL, $y = g^x$ in $(\mathbb{G}, g, q)$
- CDH, $g^x, g^y, params \rightarrow g^{xy}$
- DDH, $(params, g^x, g^y, g^{xy}) \approx_c (params, g^x, g^y, g^z)$ with $x, y, z \leftarrow \$\mathbb{Z}_q$



→ The security of KE protocol requires DL, because if DL is easy, Eve can compute x (or y) and so she can compute the key.

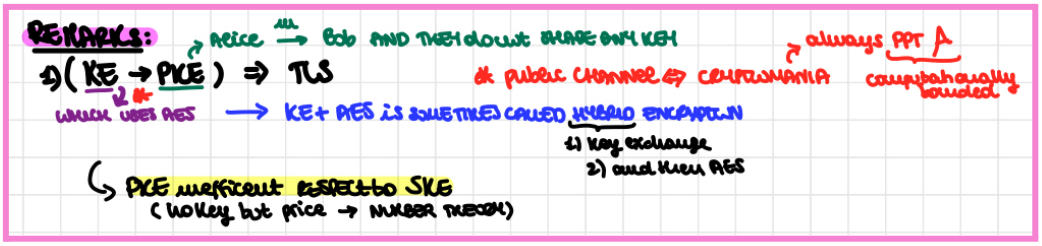
→ Under CDH assumption, the protocol is secure (secure means that the adversary cannot compute the key, it is like a tautology). But CDH just guarantees that the entire key cannot be computed, maybe it could be easy to compute a part of it. So we need DDH!

→ DDH says that the key seems random, which means the protocol is secure.

Doing crypto from DL is usually harder than doing it from DDH. In this course the majority of things that we are gonna see are from DDH.

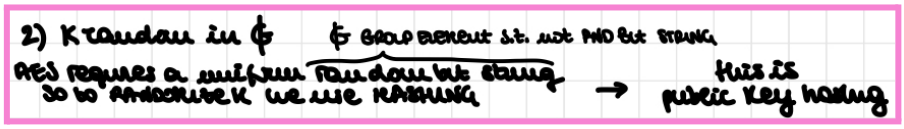
(•) REMARKS:

1. KE ⇒ PKE (TLS)



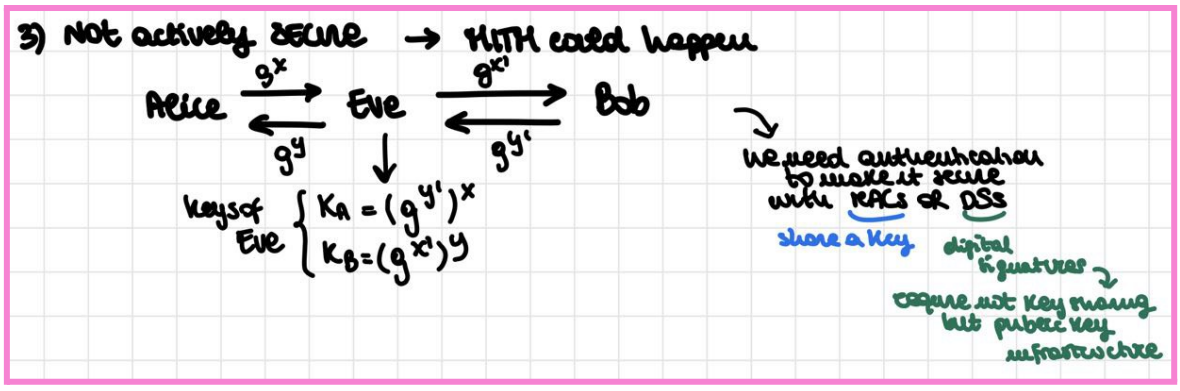
Key exchange implies public key encryption: cryptomania is the world where Alice and Bob can exchange keys over a public channel. Essentially in TLS, KE ⇒ PKE: there are the client and the server that exchange a secret key and then you can use AES (KE+AES, sometimes called hybrid encryption). So TLS could also be based directly on public key encryption, which means that Alice just sends a single message and they do not share any key.

2. Key k random in $\mathbb{G}$



The key k random in $\mathbb{G}$, but we need something random as a bit string, not as a group element. So what we do is to hash the key (hash g^{xy})!

3. Not actively secure! (Either MACs or DSs)



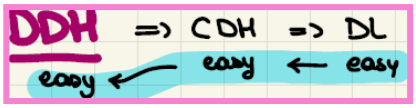
.

A Man In The Middle (MITM) attack can break it! This means that if there is no formal authentication, Eve can do a trivial MITM attack where Alice and Bob do the protocol and they are convinced that they are sharing the keys between them (but actually not). Not secure in the presence of an active adversary. How do you make it actively secure? you need to put the authentication! Two ways:

- MAC (in order to do MAC you need to share a key → circular problem);
- Digital Signatures, DS (do not require to share a key, but require other assumptions).

(•) Implications of the assumptions:

- DDH ⇒ CDH ⇒ DL



→ If you can compute g^{xy} (i.e. CDH is easy), you can also distinguish it from random (DDH is easy). If DL is easy, CDH is easy.

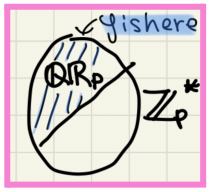
- DL⇒? CDH

→ Unknown. True in some restricted model of computation

- DL ⇏ DDH, CDH ⇏ DDH

→ $\exists \mathbb{G}\ s.t.$ DL/CDH balanced to be hard in $\mathbb{G}$, but DDH easy in $\mathbb{G}$. In fact $\mathbb{G} = \mathbb{Z}_p^*$.

To prove the last point. let's consider squaring ($\mathbb{QR}_p$ is the group of squares in $\mathbb{Z}_p^*$, quadratic residues): $\mathbb{QR}_p = \{y: y = x^2 \bmod p \ \ for\ x \epsilon \mathbb{Z}_p^*\} = \{y: y = g^z\ with\ even\ z\}$. Possiamo testare se $y \epsilon \mathbb{QR}_p$, controllando $y^{(p-1)/2} \equiv 1 \bmod p$. Perché funziona?



- Infatti se $y = g^{2z'}$ per qualche z',
  allora $y^{(p-1)/2} = g^{2z' * (p-1)/2} = g^{z'(p-1)} = 1 \bmod p$.
- Se invece $y = g^{2z'+1}$ (cioè y non è un quadrato, square),
  allora $y^{(p-1)/2} = g^{(2z'+1) * (p-1)/2} = g^{z'(p-1)} * g^{(p-1)/2} \neq 1 \bmod p$.

[(•) Tutto ciò che è più piccolo di $p - 1$ è sicuro diverso da 1 quindi $g^{(p-1)/2} \neq 1 \bmod p$]

(Recall: Fermat Little theorem $\mathbb{G} = \mathbb{Z}_p^* = \{g^0, g^1, .., g^{p-2}\}$, $p - 1$ is the order, $g^{p-1} = 1 \bmod p$, vedi Thm 27).

[1]: B.3 (Residui quadratici)

Residui quadratici modulo p. Diremo che $a \epsilon \mathbb{Z}_p^*$ è un residuo quadratico (quadratic residue) modulo p, se esiste $b \epsilon \mathbb{Z}_p^*$ tale che $a \equiv b^2 \bmod p$. L'insieme dei residui quadratici modulo p si indica con $\mathbb{QR}_p = \{a \epsilon \mathbb{Z}_p^* : a = b^2 \bmod p,\ per\ qualche\ b \epsilon \mathbb{Z}_p^*\}$.

Quante radici quadrate ha un residuo quadratico modulo p?

Lemma B.14 (Radici quadrate in QRp). Ogni elemento $a \epsilon \mathbb{QR}_p$ ha esattamente due radici quadrate modulo p. Dimostrazione (...)

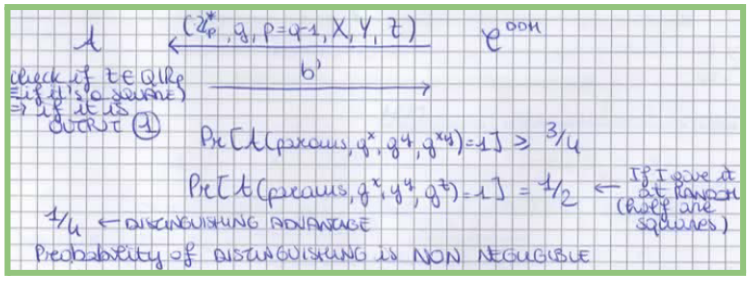
Lemma B.15 (Cardinalità di QRp). Per ogni primo p, si ha $\#\mathbb{QR}_p = (p - 1)/2$. Dimostrazione (...)

Lemma B.16 (Criterio di Eulero). Per ogni primo p, abbiamo che $a \epsilon \mathbb{QR}_p$ se e solo se $a^{(p-1)/2} \equiv 1 \ (mod\ p)$. Dimostrazione (...)

Adesso possiamo testare se un elemento è un quadrato: quando $g^{xy} \epsilon \mathbb{QR}_p$ è un quadrato (square)? Se o $g^x \epsilon \mathbb{QR}_p$ oppure $g^y \epsilon \mathbb{QR}_p$ (the power must be even in order to be a square).

Mostriamo adesso l'attacco (in figura) in cui l'avversario verifica se $Z \epsilon \mathbb{QR}_p$:

if $(Z^{(p-1)/2} \equiv 1 \bmod p) \rightarrow b' = 1$;
else $\rightarrow b' = 0$.



Analizziamo quindi:

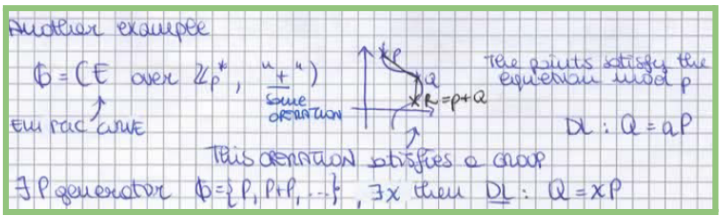
- se $Z = g^{xy}$, allora la probabilità sarà maggiore di $3/4$;
- se $Z = g^z$, allora sarà di $1/2$.

La probabilità di distinguere fra i due non è negligible (distinguishing advantage $= 1/4$):

(•) $|Pr[Eve(params, g^x, g^y, g^{xy}) = 1] - Pr[Eve(params, g^x, g^y, g^z) = 1]| = |\frac{3}{4} - \frac{1}{2}| = \frac{1}{4}$.

SOLUTION. Sia $\mathbb{G} = \mathbb{QR}_p$, prendiamo un p tale che $\frac{p-1}{2} = q$ (quindi $p = 2q + 1$) con p, q primi. Adesso ogni cosa è un quadrato (the attack does not hold anymore): not exist an algorithm that breaks DDH without solving DL.

Another example. DDH hard in $\mathbb{QR}_p$ or also in Elliptic curve groups (we won't study these).



Lecture 15

[1]: 6.1, 6.2 [2]: 11.1, 11.2, 11.5, 8.2, 8.4, 13.1 [7]: 4.3, 4.4, 4.5, 5.1

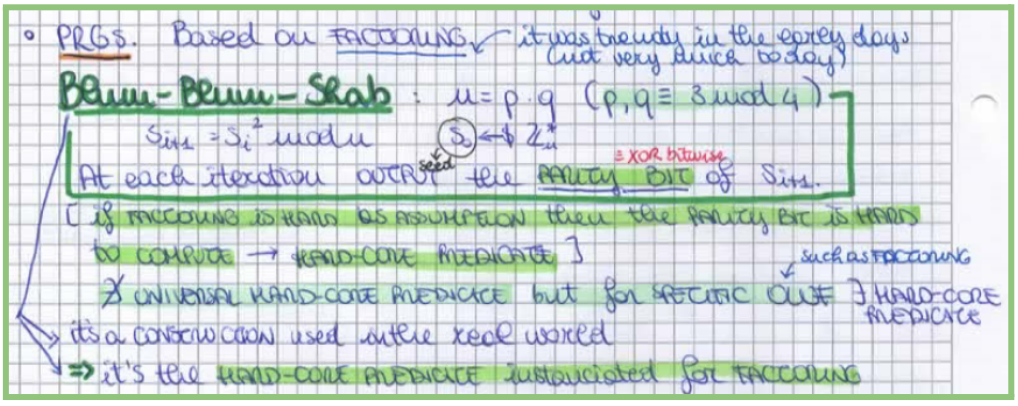
- ➢ Simple number-theoretic constructions: OWPs from DL, PRGs and PRFs from DDH (Naor-Reingold), collision-resistant hash functions from DL.
- ➢ Public-Key Encryption (PKE): Syntax and CPA security. PKE from Tradpdoor Permutations (TPDs).
- ➢ RSA Trapdoor permutation and the RSA assumption.

Simple number-theoretic constructions

Number Theoretical assumptions (Factoring, CDH, DL, DDH) could be broken by any quantum computer (still not constructed). If a quantum computer exists, what do we do and which assumptions will hold? → LPN (Learning Parity with Noise) and LWE (Learning With Error) assumptions. Number Theoretical assumptions imply PRGs, PRFs, CRHs, PKE and signatures.

(CRYPTO 2022)
Nel 2022, PRG from factoring non fatto. Blum integers citati nella Lecture 16.

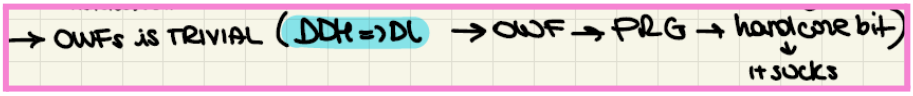
PRG from factoring



OWFs, PRGs and PRFs from DDH

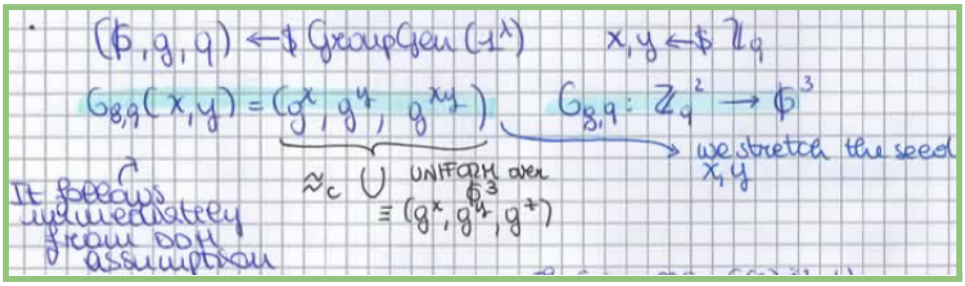
Dalle Diffie Hellman (DDH) assumptions è possibile fare ogni cosa (OWFs, PRGs, PRFs [→ MAC, CCA security]). Assume DDH is hard. Then it is easy to do everything in minicrypt.

- OWFs



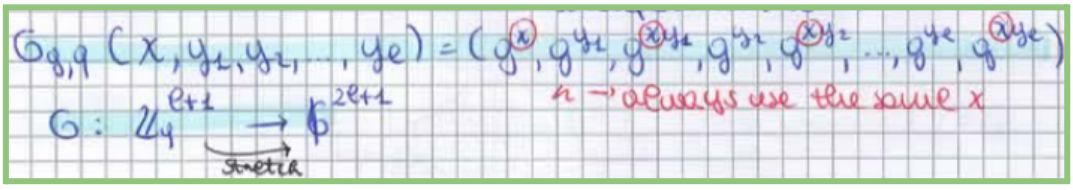
- PRGs

Una semplice costruzione di PRG da DDH è →



(•) So from DDH assumption we have that $(g^x, g^y, g^{xy}) \approx_c (g^x, g^y, g^z) \leftarrow \$\mathbb{G}^3$ (uniform random over $\mathbb{G}^3$), so $G_{g,q}$ is a PRG. It would be trivial improve the stretch repeating the construction of $G_{g,q}(x,y)$ by picking different x_s and y_s to have a longer output. But in this way also the seed is too long ($G_{g,q}(x_1,\dots,x_z,y_1,\dots,y_z) = (g^{x_1}, g^{y_1}, g^{x_1 y_1}, g^{x_2}, g^{y_2}, g^{x_2 y_2})$)!

Possiamo migliorare lo stretch (seed too long)? Yes →



The trick is to use the same x: I pick only the y_s, but only one x!

(CRYPTO 2022)

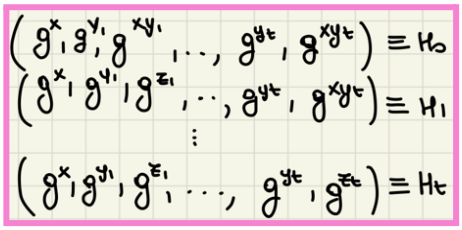
Nel 2022, enunciato come teorema [Thm D] il fatto che DDH implica una PRG $G_{g,q}$.

- Thm D - DDH → $G_{g,q}$ is a PRG

Nel 2022, dimostrazione Thm 28 lasciata come esercizio.

Enunciamo un teorema per cui la PRG definita sopra è sicura sotto le DDH assumptions, per $t = poly(\lambda)$ [Thm 28 - DDH → $G^t_{g,q}$ is a PRG for $t = poly(\lambda)$].

(•) (Proof schetch) È possibile dimostrare il teorema con hybrid argument. Definiamo gli esperimenti ibridi (in figura) e proviamo che sono tutti computationally close. We show that $H_0 \approx_c H_1$: we take X, Y, Z as the first three elements and we pick $y_2, y_3, \dots, y_t$; then the challenge is $(X, Y, Z, g^{y_2}, X^{y_2}, \dots, g^{y_t}, X^{y_t})$ and we can prove that we cannot distinguish if it is H_0 or H_1. Then we can do for all others until $H_0 \approx_c H_t$ that means that $G^t_{g,q}$ is a PRG.

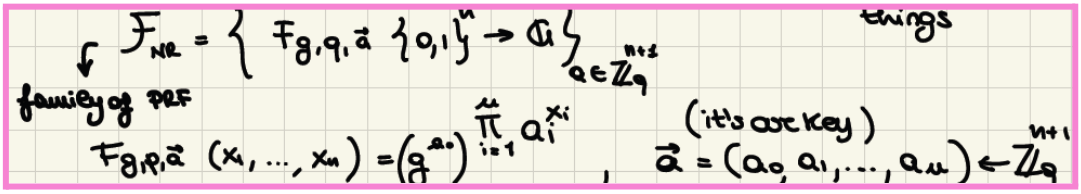


Exercise (+ idea). Prove the theorem by reduction to DDH.

- Thm 28 - DDH → $G_{g,q}^{t}$ is a PRG for $t = poly(\lambda)$ (+ proof as Exercise)

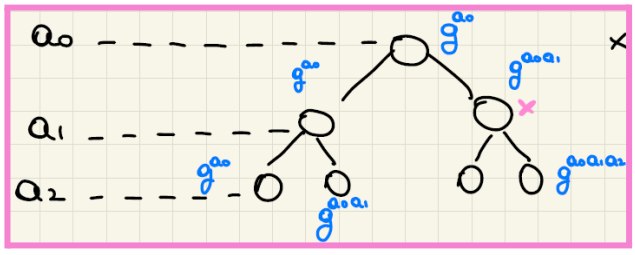
- PRF

(•) If I can do PRFs, then I have secure key encryption, MACs VIL, CBC, … everything!
Definiamo la Naor-Reingold Construction $\mathcal{F}_{NR}$. This function takes n bits and output a group element. ((•) NOTA: $F_{\vec{a}} = F_{g,q,\vec{a}}$)



(•) In the exponent $\Pi_{i=1}^{n} a_i^{x_i}$, I multiply only as with $x_i = 1$ (if $x_i = 0$ then I have $a_i^0 = 1$). So I'm using x to choose which element of $\vec{a}$ put in the exponent.

È possibile dimostrare che $\mathcal{F}_{NR}$ è una PRF sotto le DDH assumptions [Thm 29 - $\mathcal{F}_{NR}$ is a PRF family under DDH]. (INTUITION) Può essere vista come un caso speciale di GGM (theoretical construction that goes from PRGs to PRFs, vedi fine Lecture 7). Consideriamo quindi un GGM con questa PRG $G^{q,g,a}(g^b) = (g^b, g^{ab}) = G_0^a(g^b), G_1^a(g^b)$.



(•) The root of the tree is g^{a_0} (g^{a_0} is alway there). Then depending on each bit of the input, we go either left or right. If the input is like $(0, 0, \ldots, 0)$, we always go left, which means that the PRG always outputs the input (g^{a_0}). If there is a 1 in the input (x), I go right which means I put the corresponding a in the exponent. The difference from the GGM is that at every layer of this tree, I use a different seed $(a_1, a_2, \ldots, a_n)$ for the G. It's not exactly the same: in GGM you have a PRG and you always use the same PRG, instead here at every layer we are using a different PRG. But it is like GGM. With DDH, we can say that the PRG of this GGM is secure, so $\mathcal{F}_{NR}$ is a secure PRF. Thanks DDH, we can skip the proof with hybrid argument.

- Thm 29 - $\mathcal{F}_{NR}$ is a PRF family under DDH (+ proof)

CRH from DL

This is an example of a compression function for the theory approach of CRH construction. We start from DL construction (discrete log):

$params = (\mathbb{G}, g, q, y)$ (e.g. $p = 2q + 1, \mathbb{G} = \mathbb{QR}_p$),

$H_{p,q,y}(x_1, x_2) = g^{x_1} y^{x_2}$ $(2\lambda \to \lambda)$, $y \leftarrow \$\mathbb{G}$.

Now let's define a theorem which shows that this construction is CRH by starting from the assumption that it's DL (Thm E). Dimostriamo che la funzione definita sia CRH. Assume that exists an adversary that finds a collision, then you can break DL.

Siano $(x_1, x_{2,}) \in \mathbb{Z}_q^2$ e $(x_1', x_{2,}') \in \mathbb{Z}_q^2$ tali che ci sia una collisione, i.e $g^{x_1} y^{x_2} = g^{x_1'} y^{x_2'} \; mod\, p$.

Da qui possiamo ricavarci che: $g^{x_1} y^{x_2} = g^{x_1'} y^{x_2'} \; mod\, p \Rightarrow g^{(x_1 - x_1')} = y^{(x_2 - x_2')} \; mod\, p$. Possiamo affermare adesso che wlog (without loss of generality) $x_2 - x_2' \neq 0$, altrimenti x_1 sarebbe uguale a x_1' e non ci sarebbe collisione (poiché per avere una collisione $(x_1, x_{2,})$ deve essere diverso da $(x_1', x_{2,}')$).

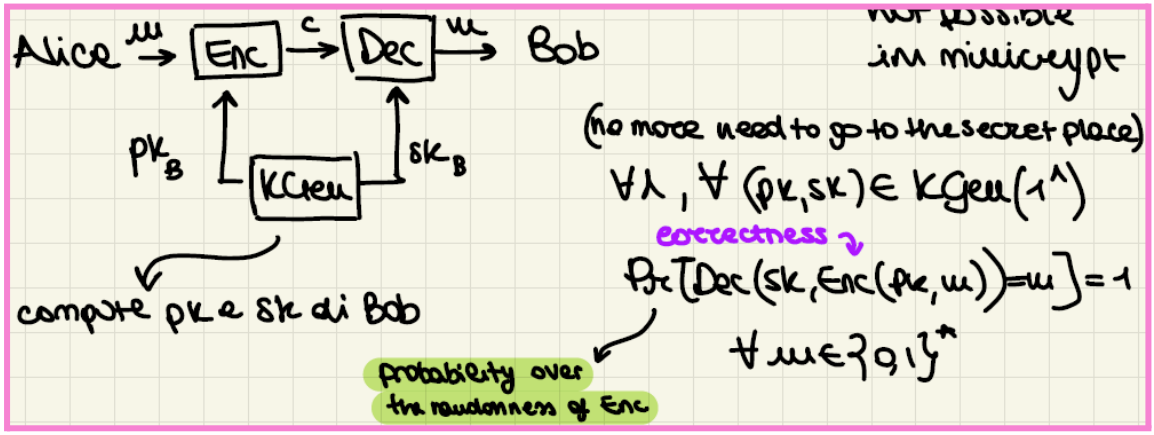
Di conseguenza $y = g^{(x_1 - x_1')(x_2' - x_2)^{-1}} \; mod\, p$: this breaks DL (Note: $(x_2' - x_2) \in \mathbb{Z}_q$ and is different from 0, so you can invert it $\rightarrow (x_2' - x_2)^{-1}$ exists).

(CRYPTO 2022)
Nel 2022, la resistenza alle collisioni in DL della costruzione viene enunciata come teorema (e successivamente dimostrata). Nel 2021, si procede direttamente con la dimostrazione.

- Thm E - Assuming DL, above construction is CRH (+ proof)

Public key encryption

(•) PKE is another way to do message confidentiality, but it's only possible in cryptomania. The idea is that Alice to send a single message to Bob uses a $KGen$ that takes as input the security parameter (1^λ) (in questo caso la chiave non è più uniformly random, ma è generata a partire da un algoritmo). $KGen$ is computing pk and sk of Bob (pk_B e sk_B in figura) and Alice need to know the pk of Bob.



We have that the probability to decrypt m with sk given $Enc(pk, m)$ is 1 over the randomness of the Enc algorithm. This means Enc is not a deterministic algorithm: whenever I encrypt it I will get a different ciphertext ($c^* \leftarrow \$Enc(pk, m_b) \Rightarrow r \leftarrow \$\{0,1\}^*, Enc(pk, m_b; r)$).

RECALL. In SKE (Secret Key Encryption) $Enc = (r, F_k(r) \oplus m), r \leftarrow \$\{0,1\}^n$.

SUBTLETY. With Public Key Encryption, there is no more need to go to the secret place (Alice only has to know the public key). How does Alice know that pk_b really is the public key of Bob? Alice needs to be sure it's Bob's key (Eve could send Alice the wrong key and then read everything Alice sends, and encrypt). Technical detail: we don't want to share a secret but how can we be sure of the authentication of the public key? How to solve this? Require

setup! Need to authenticate pk_b using signatures! For now we assume that Alice knows the right public key (we'll analyze later this subtlety [(•) quando vedremo le digital signatures → le public key infrastructure (PKI) le usano il che permette PKE]).

(CRYPTO 2022)
Nel 2022, RSA è stato fatto prima di TDP.

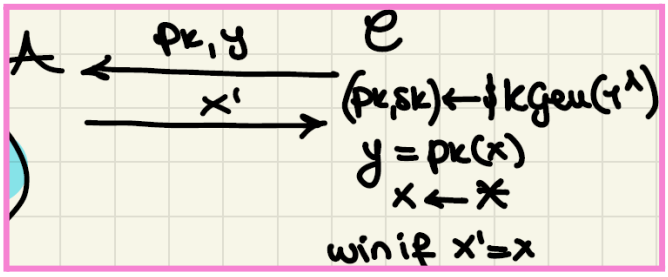
TDP (Trapdoor Permutation)

(•) QUESTION. What is the "minimal" assumption to imply PKE? In minicrypt the minimum assumption was OWF. If you have any OWF, you can construct all the primitives in the symmetric key world. And now the question is if there exists a corresponding assumption that suffices for constructing all the primitives in cryptomania. OWF is not the answer (OWF⇏PKE, FACT). Second, we have said that RSA implies factoring (vedi più avanti) but what about the opposite direction? In Minicrypt, if I know a secret key I can invert it. In this sense when I know the secret key I have a OWP: $f_{sk}^{-1}(y) = y^d \bmod n$. The fact that the inverse exists doesn't necessarily imply that its computation is efficient. What is the answer to this? TDP: Trapdoor Permutation.

(CRYPTO 2022)
Nel 2022, TDP enunciata come definizione.

- Def B - Trapdoor permutation (TDP)

TDP sono OWP con una trapdoor. Definiamo TDP una primitiva $\Pi = (KGen, f, f^{-1})$ tale che per ogni $\lambda \in \mathbb{N}$ e per ogni $(pk, sk) \leftarrow KGen(1^\lambda)$ allora $f_{pk}: \mathfrak{X}_{pk} \to \mathfrak{X}_{pk}$ è una permutazione in $\mathfrak{X}$ ($f_{sk}^{-1}: \mathfrak{X}_{pk} \to \mathfrak{X}_{pk}$). Infatti per ogni $x \in \mathfrak{X}_{pk}$ (space of permutation may/could depend on the public key) e per ogni coppia valida (pk, sk), $f_{sk}^{-1}(f_{pk}(x)) = x$; sk è chiamata trapdoor e per un avversario, che non conosce sk, f_{pk} è una OWP. Definiamo quindi un game (in figura a destra) in cui l'avversario vince se riesce a computare $x' = x$, con x computato randomicamente dal challenger su $\mathfrak{X}_{pk'}$ a partire dalla public key pk e dall'output y di $f_{pk}(x)$ ($\to y = f_{pk}(x)$).



It's a OWF with a trapdoor:

- if you know only the public key → it's one way
- but if you know the secret key you can also invert it.

Is this secure? NO. It is not CPA secure, because this is deterministic (same message, same ciphertext), so the adversary can break it. So it is not secure as a PKE (What is a secure PKE? → CPA + CCA).

(•) In this case, differently from minicrypt, the adversary should know the public key pk. Furthermore, you have that the permutation function f_{pk} necessary should be a One-Way (OW) and not a OWP. What does it mean? If it's OWP you know the secret key and you can also invert it but in this case it's sufficient to have the public key. This behavior is the same as the RSA assumption (2022 fatta prima). In that case you've as TDP y as the $f_{pk}(Enc(pk, m))$ and $x = f_{sk}^{-1}(Dec(sk, c))$. Then RSA assumption and TDP are equivalent. This implies that textbook RSA (vedi dopo) is a TDP under RSA assumption.

CPA security for PKE

Now, how to define security?

1. Hard to recover sk from pk. BAD
   What if $Enc(pk, sk)$ reveals the first bit of m or something about m in general?
2. Hard to recover m from c. BAD
   What if you recover not the whole message but a part of it?
3. CPA security. GOOD

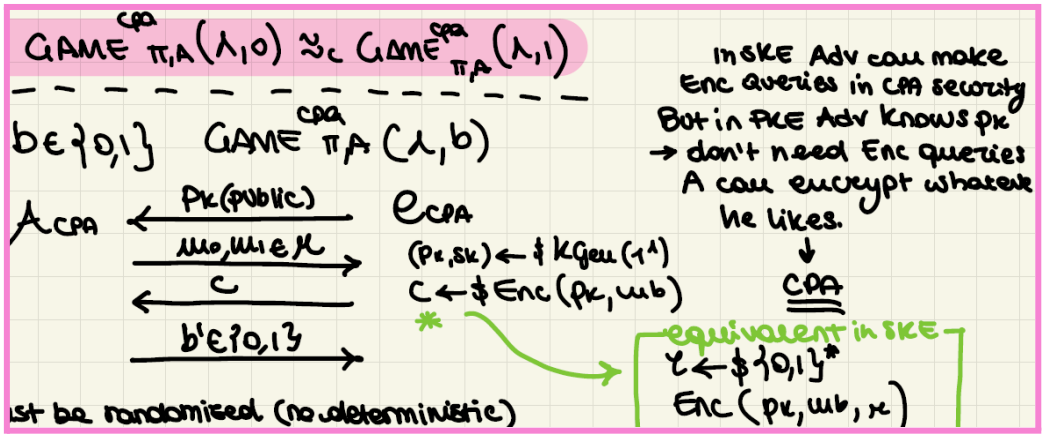
Our security notion for PKE will be CPA-security.

(CRYPTO 2022)

Nel 2022, CPA-security per PKE enunciata come definizione.

- Def C - CPA-security for PKE

Traduciamo in PKE quello che abbiamo visto di CPA security per SKE. $\Pi = (KGen, Enc, Dec)$ è CPA secure se $Game_{\Pi,A}^{CPA}(\lambda, 0) \approx_c Game_{\Pi,A}^{CPA}(\lambda, 1)$, all'interno del game in figura.



(•) La differenza tra SKE e PKE è che SKE l'avversario fa le encryption queries perché non conosce la chiave mentre qui la chiave è pubblica e quindi conoscendo l'encryption algorithm non ha bisogno di fare encryption queries.

Observations:

- Enc deve essere randomizzato (no deterministic encryption), perchè l'avversario potrebbe distinguere altrimenti i due game (two times encryption of same message → same cipher). So TDP (that is not randomized) is not CPA secure. TDP is good

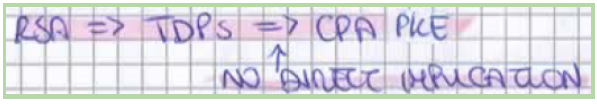
because it allows us to encrypt with public key and decrypt only with secret key, but the way we saw it, it's not CPA secure.

- Given pk, you can't compute sk.
- Looking at c, you can't recover m_0, m_1.

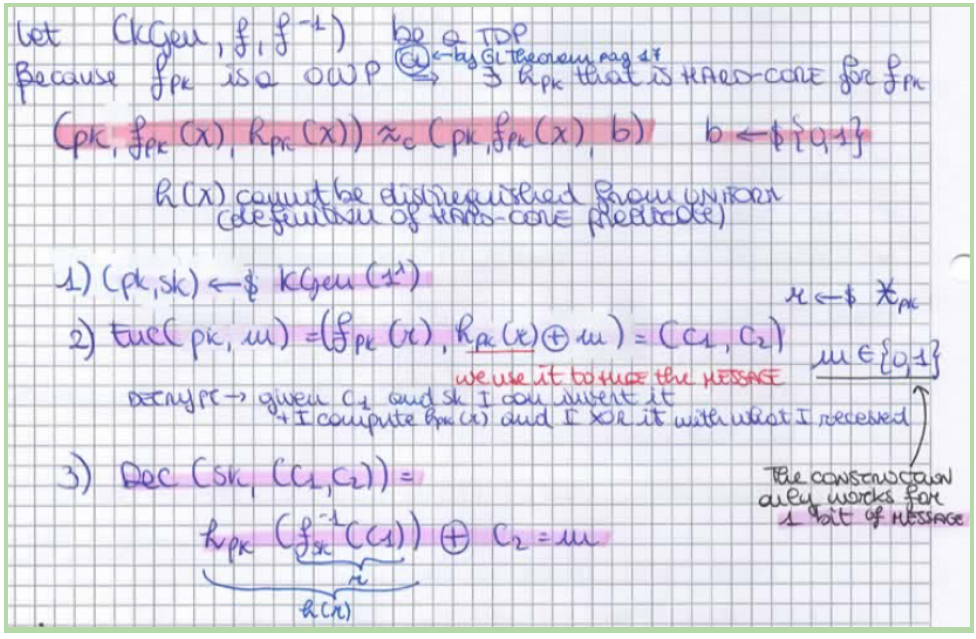
(•) GOAL. How to construct CPA public encryption? We will see RSA (RSA '78 insecure respect to RSA '84 with CPA), ElGamal, TDP, etc. Then we will talk about CCA public encryption.

CPA PKE from TDP

(•) Next question. Does a TDP directly imply PKE? Well $Enc(pk, m) = f_{pk}(m)$ is not CPA secure (it is deterministic). This motivates why RSA is not CPA-secure.



TDP non è direttamente CPA secure, ma dato TDP, CPA PKE può essere costruito.
Definiamo quindi un teorema per cui TDP implica una CPA PKE [Thm 30 - TDP⇒CPA PKE] (Dimostrazione 2021)



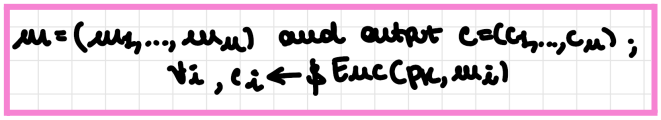
- Thm 30 - TDP⇒CPA PKE (+ proof)

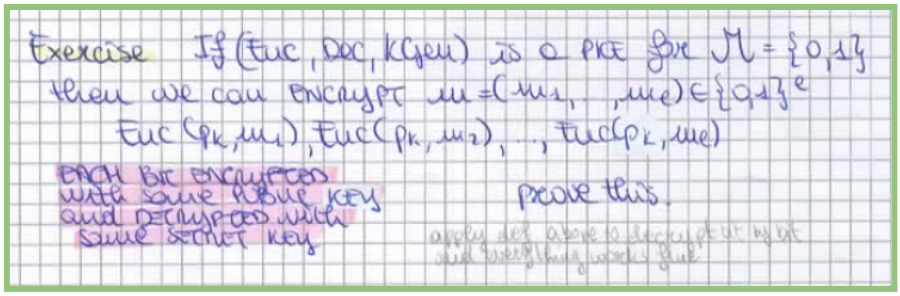
(CRYPTO 2022)
Nel 2022, la dimostrazione del Thm 30 è presentata in maniera più semplice.
(Proof) Observation. In CRYPTOMANIA: TDP⇒PKE↔KE.
We start the proof by assuming a construction from $\mathcal{M} = \{0, 1\}$.
Let $(KGen, f, f^{-1})$ be a TDP and let h_{pk} be HARD-CORE for f_{pk} (h_{pk} is what you cannot compute of x given y). Then, we can start by constructing $Enc(pk, m) = (f_{pk}(r), r \oplus m)$.
In this way, $r \oplus m$ is OTP (information theoretically secure) then Bob knows sk (he can decrypt). But what reveals $f_{pk}(r)$ (i.e. c_1) of r? It could reveal the first bit or something else

(not all r for the definition of OWP). For this reason, this construction is not CPA-secure in general.
How do you fix it? $Enc(pk, m\epsilon\{0,1\}) = (f_{pk}(r),\ h_{pk}(r) \oplus m)$, with $r \leftarrow \$\mathfrak{X}_{pk}$.
Because the hard-core you ensure the hardness of the message. In fact, the hard-core is uniform, then the xor between them is uniform. We can do a reduction and prove it.
Note. If the function f_{pk} is put to be something else (not the generic function, but maybe an RSA), then hard-core could be longer (and not only one bit, GL one bit). However, this is not practical.

(•) This theorem is very important because it shows that the minimal assumption to do PKE is TDP. This $Enc(pk, m\epsilon\{0,1\}) = (f_{pk}(r),\ h_{pk}(r) \oplus m)$ is an alternative encryption scheme for RSA. under RSA assumption, I can do this and I get a PKE CPA secure.

How to do PKE for poly many bits? If m is many bits, PKE can be done e.g. with a random oracle or encrypt each bit individually. In practice this construction is useless; other assumptions are used (they immediately encrypt long messages). Theoretically, TDP is the only assumption/construction we need.
Exercise (+ idea). PKE for one bit ⇒ PKE for many bits. (•) You need to apply the definition of the proof above to decrypt bit by bit in this way:





(•) Observation. Confronto con hash function: any CRH doesn't need to hide the input; with any hash function you could always modify it to have that the output is enlarged then in general you can do extension attacks.

RSA (Rivest–Shamir–Adleman)

(•) Textbook RSA (1978) is insecure (no CPA security). It's based on factoring conjecture (given n, it's hard to compute p and q). By the fact that TDP⇒CPA PKE, FACTORING⇒TDP.
$KGen(1^{\lambda})$: $n = p * q$, con p, q primes of λ bits. La nostra OWF è $p * q$.
Let fix some $e, d\ s.t.\ e * d \equiv 1\ mod\ \varphi(n)$ with $\varphi(n) = (p-1)(q-1)$.
$\varphi(n)$ può essere provato essere pari alla cardinalità degli elementi $mod\ n$ che sono invertibili (set dei messaggi coincide con il set $\mathbb{Z}_n^*$, $\mathbb{Z}_n^* = \mathcal{M}$). e e d invece sono rispettivamente

l'encryption e il decryption exponent; $e = 3$ è una buona scelta (implica che ci sono tanti zeri, quindi una encryption più veloce), con d l'inverso di e.
Then we have: $pk = (n, e)$, $sk = (d, p, q, n)$ as the public and secret keys, and the following algorithms $Enc(pk, m)$: $c = m^e \bmod n$, $Dec(sk, c)$: $m = c^d \bmod n$.

OBSERVATION:

- it is not CPA secure (it is deterministic)
- it needs the hardness of FACTORING (at least → we'll show something else in addition is needed) because security is related to factoring. In fact if I can find p, q then the encryption is not one-way.
- it satisfies correctness, because

  $c^d \equiv (m^e)^d \equiv m^{e*d} \equiv m^{1 \bmod \varphi(n)} \equiv m^{t*\varphi(n)+1} \equiv m^1 * m^{t*\varphi(n)} = m(m^{\varphi(n)})^t \bmod n$

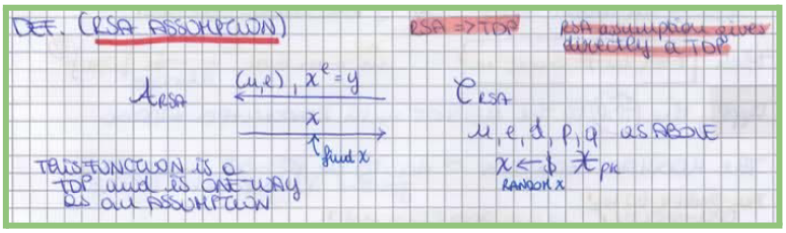
  che per il teorema di Eulero $m(m^{\varphi(n)})^t \equiv m * 1 = m \bmod n$.

(•) $m^{1 \bmod \varphi(n)} \equiv m^{t*\varphi(n)+1}$ it's valid because 1 is the reminder between $e * d$ and $\varphi(n)$ so that $e * d$ is a multiple of $\varphi(n)$ [t è il quoziente].

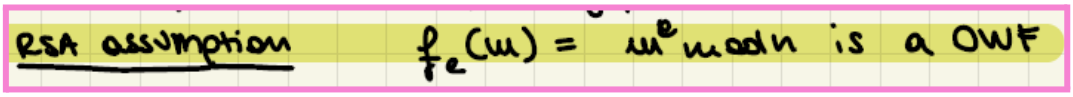
RSA assumption

(CRYPTO 2022)
Nel 2022, RSA assumption non enunciata come definizione e game relativo all'assunzione (fine Lecture 15 e inizio Lecture 16) non fatto.

Definiamo la RSA assumption [Def 23 - RSA Assumption] che afferma che RSA⇒TDP.



(•) RSA assumption: $f_e(m) = m^e \bmod n$ is a OWF.



(•) RSA assumption is the only thing you can prove it's secure. Abbiamo le seguente relazione: RSA⇒Factoring, Factoring⇒?RSA.
Per questo motivo per provare che RSA (PKCS #1.5) è sicuro (CPA secure) ci serviamo di RSA assumption che implica factoring.

- Def 23 - RSA Assumption

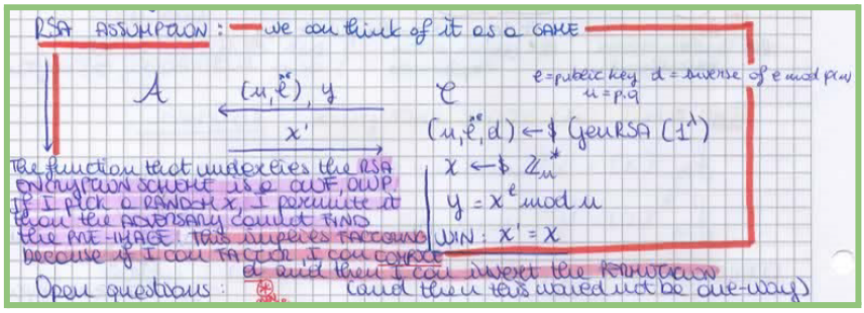
Lecture 16

[1]: B, 6.2 [2]: 11.5, 13.5 [7]: 5.1, 5.2

➢ RSA encryption (with mentions to PKCS #1 v1.5 and PKCS #2 v2.0). Squaring mod n: Rabin's TDP and its equivalence to Factoring.

RECAP. Nell' ultima lezione abbiamo visto che:

- (Thm 30) TDP⇒CPA PKE (only for 1-bit messages)
- (Def 23) RSA⇒TDP (RSA assumption gives directly a TDP)



OPEN QUESTIONS

1. PKE from factoring?
2. Can we get a more efficient PKE (that works for more than one bit)?
3. Other assumptions (DDH)? CCA security?

PKCS #1 v1.5 and PKCS #2 v2.0

2. Can we get a more efficient PKE (that works for more than one bit)?
How to make it secure? → Standard PKCS #1.5.

L'idea è avere $c = (\hat{m})^e \ mod\ n$, con $\hat{m} = r||m$ e $r \leftarrow \$\{0,1\}^l$ (randomized padding). So instead of encrypting directly m, you pick random r of length l and you encrypt $r||m$. Quindi applichiamo TDP al messaggio con randomized padding.

(•) This gives hope, but is it CPA secure?

- if $l = O(log\ \lambda)$ then not secure (can guess r in polynomial time)
  if $l = 1$ then not secure (can guess r w.p. $1/2$)
- if $m \epsilon \{0,1\}$ can prove it CPA secure. Under which assumption? Factoring? Unknown!
  The only thing you can prove is secure is under RSA assumption.
  RSA assumption⇒Factoring. Why? Because I can invert.
  Factoring implies RSA assumption? Nobody knows! (Factoring⇒?RSA assumption)
- Other cases nobody knows.

Other important things to know…

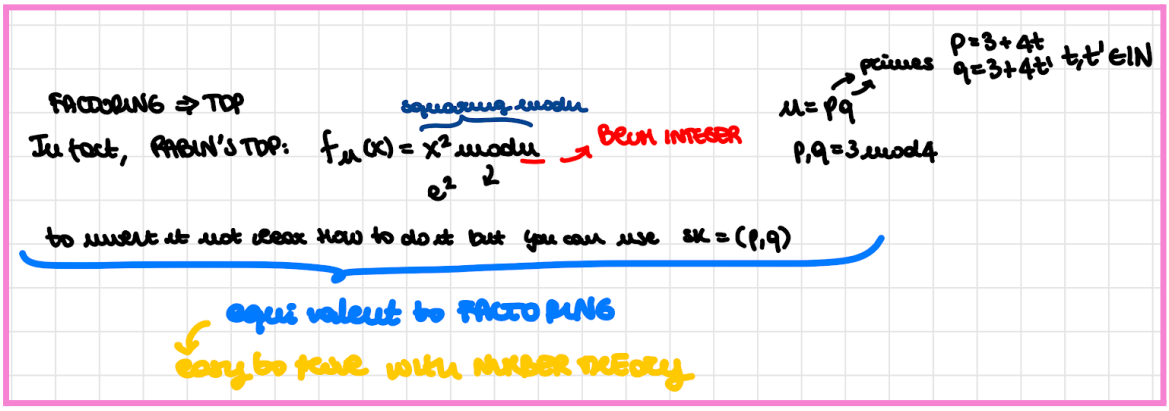
- Not CCA secure

PKCS #1.5 is not CCA secure. There are some real world attacks that happen against TLS (TLS uses RSA). A guy showed that if you want to break a ciphertext, you can flip some bits, then send it to the server and the server will decrypt it. And then will not send the message, but only tell your first bit of information about each ciphertext saying if the padding is correct or not (the padding needs to have some specific format, otherwise it is invalid). To fix this we should pick PKCS #2, also called RSA-OAEP (Optical Asymmetric Encryption Padding).

Rabin's TDP and its equivalence to Factoring

1. PKE from factoring?

(•) Factoring is an unknown assumption to prove it's secure (CPA/CCA): remember we use RSA assumption.



(•) Rabin's TDP is equivalent to Factoring.
For this particular function you can prove two things.

1. Factoring⇒Rabin's TDP: in fact, $f_n(x) = x^2 mod\ n$ (as above). (Nota, vedi Lecture 15)
2. Rabin's TDP⇒Factoring: there exists the inverse and, even is not clear how to do this, it can be inverted with $sk = (p, q)$ (if you know the factorization of n). If an adversary can invert it without p and q, then this adversary can be hard into a machine which factors n.

Now we can base PKE on factoring and not on RSA assumption.

(CRYPTO 2022)
Nel 2022, dimostrazione Rabin TDP equivalent to factoring non fatta.

Elgamal PKE and its CPA security from the DDH assumption

3. CPA/CCA PKE from DL (DDH)
(•) Luckily we know today how to get CPA/CCA PKE based on factoring, DDH, DL, etc.
Fact. Assuming factoring, then is PKE for $poly(\lambda)$ bit messages with CPA/CCA security ('2000). This can be shown by ElGamal PKE (1984) and we'll see it's CPA secure under DDH assumption (DDH⇒CPA PKE).

Let's see the scheme $\Pi = (KGen, Enc, Dec)$ of ElGamal!

- $KGen(1^\lambda)$: $params = (\mathbb{G}, g, q) \leftarrow \$GroupGen(1^\lambda)$; $x \leftarrow \$\mathbb{Z}_q$; $h = g^x$ (all operations in $\mathbb{G}$); $pk = (params, h)$; $sk = x$.
- $Enc(pk, m \in \mathbb{G})$: $c = (c_1, c_2) = (g^r, h^r * m)$; $r \leftarrow \$\mathbb{Z}_q$ (the encryption is inherently randomized)

(•) Note. The product * depends on $\mathbb{G}$. If $\mathbb{G} = \mathbb{Z}_n$ then it's $*\ \ mod\ n$. NB2. m is long many bits (not only 1) because in $\mathbb{G}$.

- $Dec(sk, (c_1, c_2))$ Output $c_2/{c_1}^x = \frac{h^r * m}{{c_1}^x} = \frac{h^r * m}{(g^x)^r}$, $g^x = h$ quindi $c_2/{c_1}^x = m$ (here, we show that Bob can decrypt)

Nella prossima lezione dimostreremo che ElGamal PKE è CPA secure sotto DDH (Thm 31).

(•) Some observation

- ElGamal is re-randomizable

Given a ciphertext, you can transform it in a new one under a freshly random r: given $(g^r, h^r * m) = (c_1, c_2)$ then $(c_1 * g^{r'}, c_2 * h^{r'}) = (g^{r+r'}, h^{r+r'} * m)$ with $r' \leftarrow \$\mathbb{Z}_q$

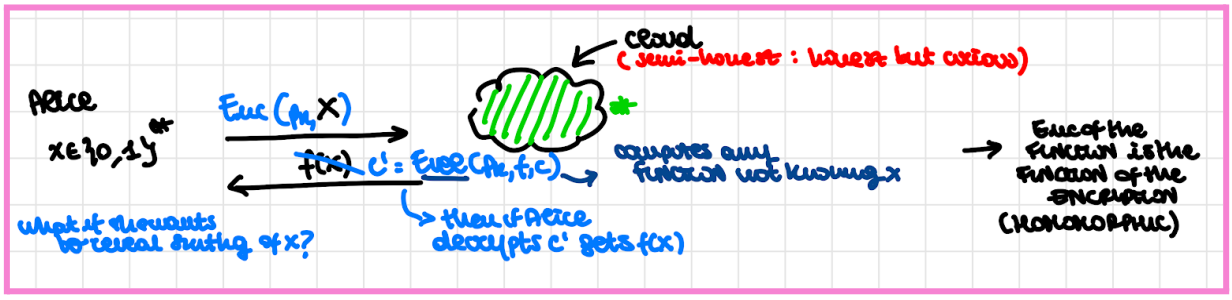
- ElGamal is homomorphic (cool property)

$c = (c_1, c_2) = (g^r, h^r * m)$, $c' = (c_1', c_2') = (g^{r'}, h^{r'} * m')$

$c'' = (c_1 * c_1', c_2 * c_2') = (g^{r+r'}, h^{r+r'}(m * m')) \equiv Enc(pk, m''; r'')$ [given r'']

An idea of Rivest is: FHE (Full Homomorphic Encryption, 1980).

Want $Enc(pk, m)$ such that it exists PT Eval for which $c \leftarrow \$Enc(pk, m)$; then $Eval(pk, f, c) = c' \equiv Enc(pk, f(m))$ $\forall f: \{0,1\}^* \rightarrow \{0,1\}^*$ and this is called outsourcing of computation.



First solution: Gentury (1910)

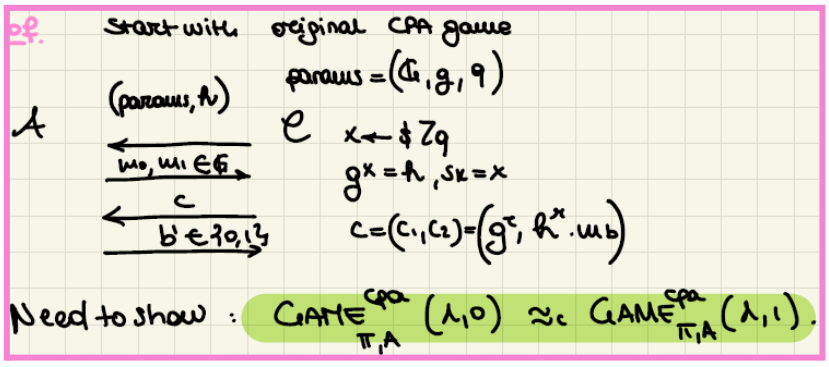
Today: possible under standard assumption.

Lecture 17

[1]: 6.3, 6.5 [2]: 11.4 [5] [7]: 5.3

- ➢ Elgamal PKE and its CPA security from the DDH assumption.
- ➢ Cramer-Shoup encryption.

Enunciamo il teorema per cui, sotto DDH, Elgamal PKE è CPA secure [Thm 31 - Under DDH, the Elgamal PKE is CPA secure]. Per la dimostrazione, partiamo dal game originale di CPA per PKE (vedi Lecture 15), utilizzando lo schema di Elgamal [(•) we're hardening m_b using h^r]; dobbiamo dimostrare che $Game^{CPA}_{\pi,A}(\lambda, 0) \approx_c Game^{CPA}_{\pi,A}(\lambda, 1)$.



L'intuizione è che A conosce $h = g^x$, $c_1 = g^r$ e $h^r = g^{xr}$. Sotto DDH, $(g^x, g^r, g^{xr}) \approx_c (g^x, g^r, g^z)$, $x, y, z \leftarrow \$\mathbb{Z}_q$ [(•) We use DDH assumption which we know implies DL].

(•) h^r is indistinguishable from random (uniform) in fact the message m is informationally theoretical hidden even if g^x and g^r are known.

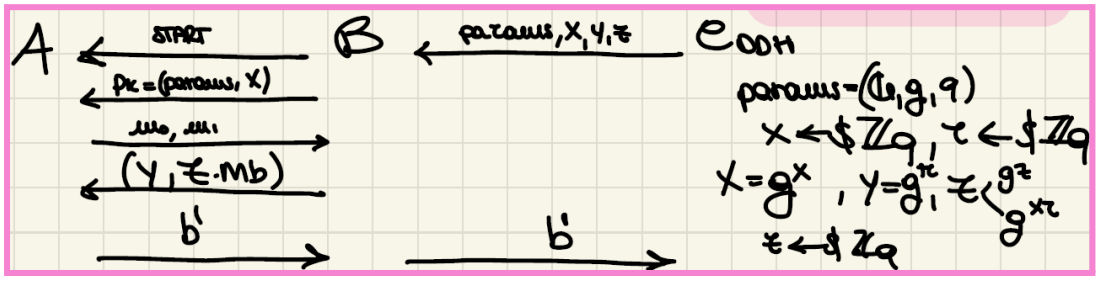
Come formalizziamo questa intuition? Consideriamo un hybrid experiment $Hyb_{\pi, A}(\lambda, b)$, dove ci sarà sempre la stessa interazione con l'avversario, ma che computa c_2 tramite g^z ($c_2 = g^z * m_b$, $z \leftarrow \$\mathbb{Z}_q$), a differenza dell'esperimento originale dove c_2 era computata tramite g^{xr} ($c_2 = h^r * m_b$). Proseguiamo la dimostrazione con due lemmi.

- [Lemma 1] $Hyb_{\pi, A}(\lambda, 0) \equiv Hyb_{\pi, A}(\lambda, 1)$

Il primo lemma è facilmente dimostrabile, dato che la distribuzione dei ciphertext è uniformemente randomica e non dipende da b.

- [Lemma 2] $Game^{CPA}_{\pi,A}(\lambda, b) \approx_c Hyb_{\pi, A}(\lambda, b)$.

Dimostriamo il secondo lemma invece tramite una riduzione (in figura sotto) a DDH (assume not there exist a PPT adversary A that violates the lemma, i.e. that can distinguish $Game^{CPA}_{\pi,A}(\lambda, b)$ from $Hyb_{\pi, A}(\lambda, b)$; then we assume exists B s.t. …).

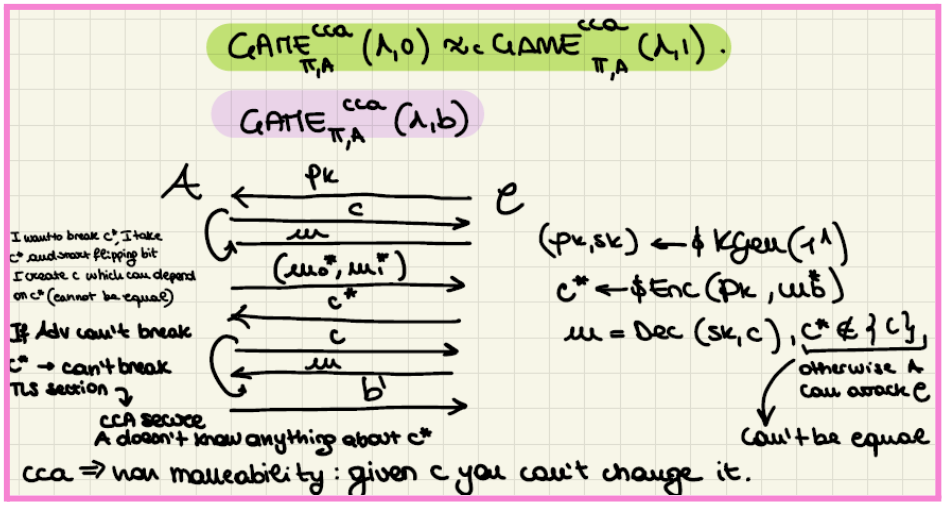


Exercise. Prove the lemma by reduction to DDH.

- Thm 31 - Under DDH, the Elgamal PKE is CPA secure (+ proof) + Lemma 1 (+ proof) + Lemma 2 (+ proof as Exercise)

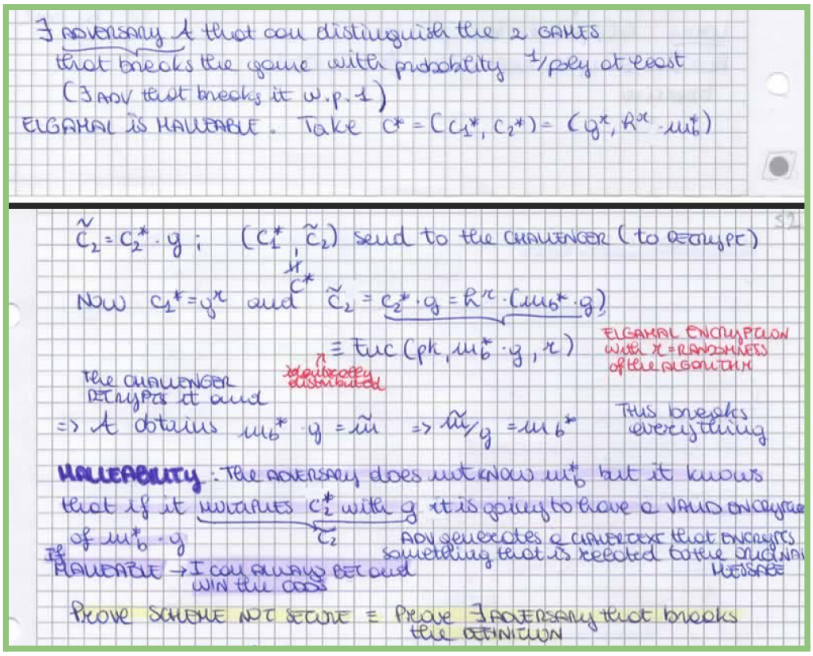
CCA security for PKE

Definiamo uno schema $\Pi = (KGen. Enc, Dec)$ CCA secure [Def 24 - CCA-security for PKE] se $Game^{CCA}_{\Pi,A}(\lambda, 0) \approx_c Game^{CCA}_{\Pi,A}(\lambda, 1)$.

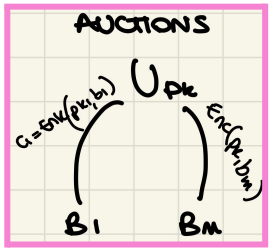


- Def 24 - CCA-security for PKE

Exercise (+ idea). Elgamal is not CCA secure



(•) CCA security is equivalent to non-malleability. It's malleable → It's homomorphic. In addition, we've also said it's re-randomizable, then we need to choose $c = (g^{r'} c_1^*, h^{r'} c_2^*)$. Otherwise, from c^* I can find $c = Enc(pk, 2m_b^*)$. [Like auctions, vedi figura a lato]



QUESTION. How to obtain CCA PKE from DDH? → Cramer-Shoup (1998)
We have two options:

1. direct construction from DDH
2. hash proof system (instantiation from DDH)

(CRYPTO 2022)
Nel 2022, si procede con la prima modalità (direct construction from DDH); hash proof system non fatto.

2022

Di seguito, le lezioni del 2022 dalle Lecture 19 & Lecture 20 del 02/12/22.

Lecture 19 & Lecture 20

[1]: 6 [5] [7]: 5.3

- ➢ Cramer-Shoup encryption.

Cramer-Shoup encryption

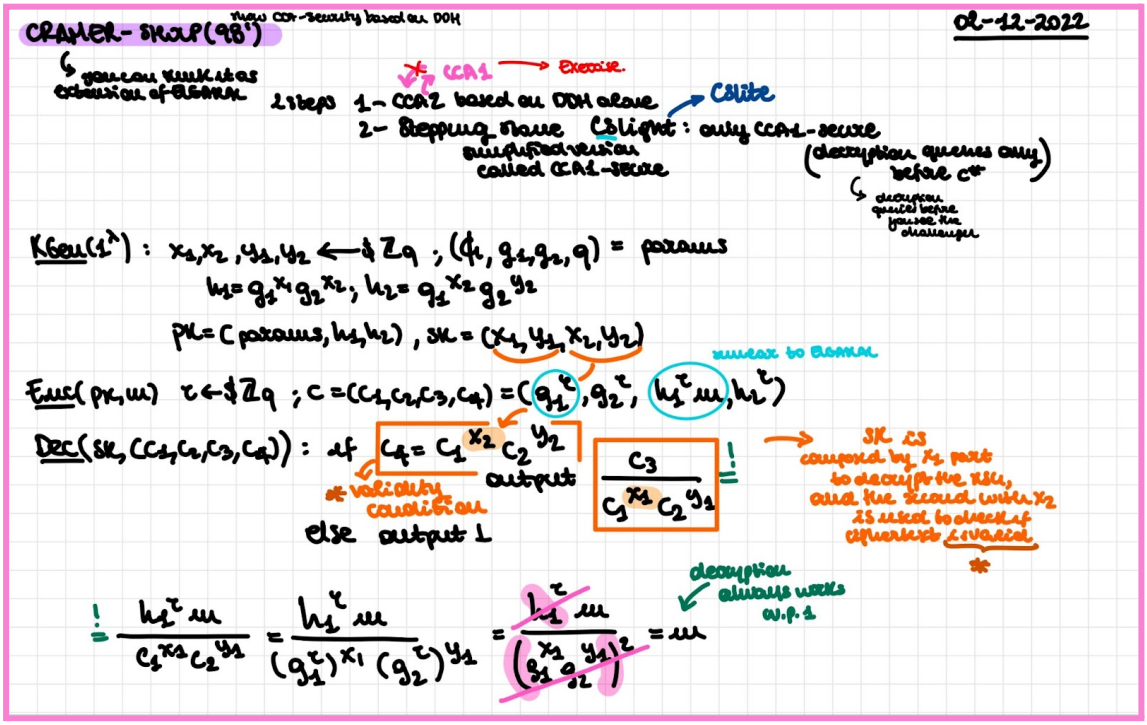
We can think Cramer-Shoup (1998) as an extension of Elgamal. We proceed with two steps:

- CCA2 based on DDH alone
- Stepping stones CSLight (CSlite): CSlite is only CCA1 secure (decryption queries only before c^*)

In CCA1 no decryption queries can be made after receiving the ciphertext, while in CCA2 decryption queries can be made for ciphertexts different from the challenge ciphertext ([7]: Definition 25 (CCA-1 for PKE) e Definition 26 (CCA-2 for PKE)).

Note. CCA2⇒CCA1, but CCA1⇏CCA2

Exercise. Prove the above note.



Cramer Shoup construction:

- $KGen(1^{\lambda})$: $x_1, x_2, y_1, y_2 \leftarrow \$\mathbb{Z}_q$; $(\mathbb{G}, g_1, g_2, q) = params$

  $h_1 = g_1^{x_1} g_2^{y_1}$, $h_2 = g_1^{x_2} g_2^{y_2}$

  $pk = (params, h_1, h_2)$, $sk = (x_1, y_1, x_2, y_2)$

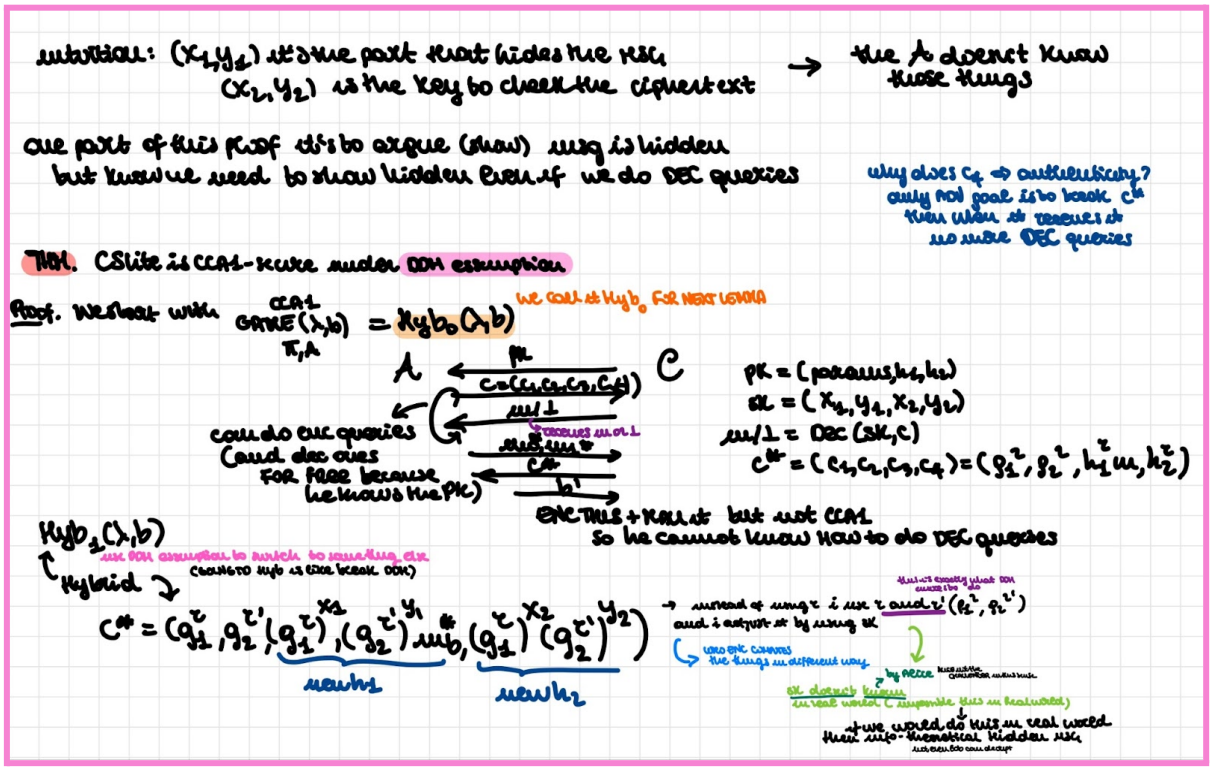
- $Enc(pk, m)$: $c = (c_1, c_2, c_3, c_4) = (g_1^r, g_2^r, h_1^r * m, h_2^r), r \leftarrow \$\mathbb{Z}_q$
- $Dec(sk, c_1, c_2, c_3, c_4)$:

  if $c_4 = c_1^{x_2} * c_2^{y_2}$

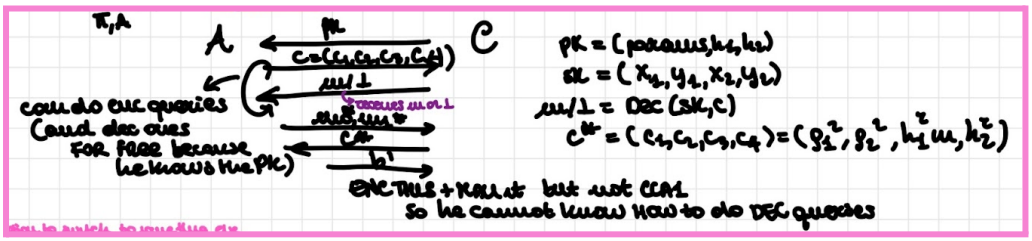
  Output $\frac{c_3}{c_1^{x_1} * c_2^{y_1}} = \frac{h_1^r * m}{c_1^{x_1} * c_2^{y_1}} = \frac{h_1^r * m}{(g_1^r)^{x_1} * (g_2^r)^{y_1}} = m$

  $[\ (g_1^r)^{x_1} * (g_2^r)^{y_1} = (g_1^{x_1})^r * (g_2^{y_1})^r = (g_1^{x_1} g_2^{y_1})^r = (h_1)^r\ ]$

  else Output $\perp$

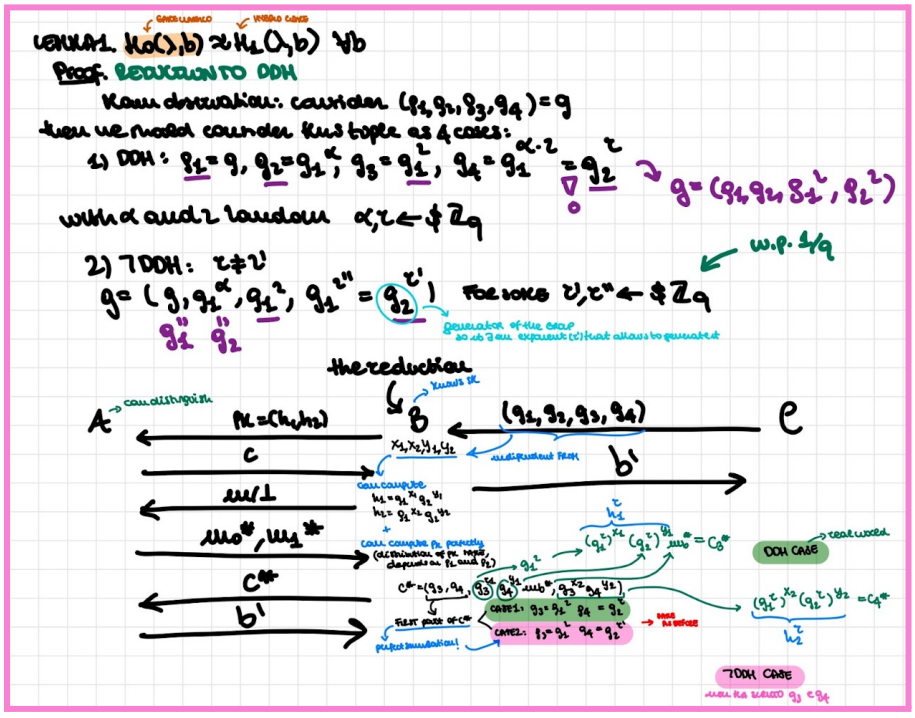


We can prove that Cramer-Shoup construction is CCA1 secure under DDH [Thm F - CSlite is CCA1 secure under DDH]; let's prove it. We start with $Game_{\Pi,A}^{CCA1}(\lambda, b)$ (in figura sotto): for the CS construction, $c^* = (c_1^*, c_2^*, c_3^*, c_4^*) = (g_1^r, g_2^r, h_1^r * m_b^*, h_2^r)$ with $r \leftarrow \$\mathbb{Z}_q$.



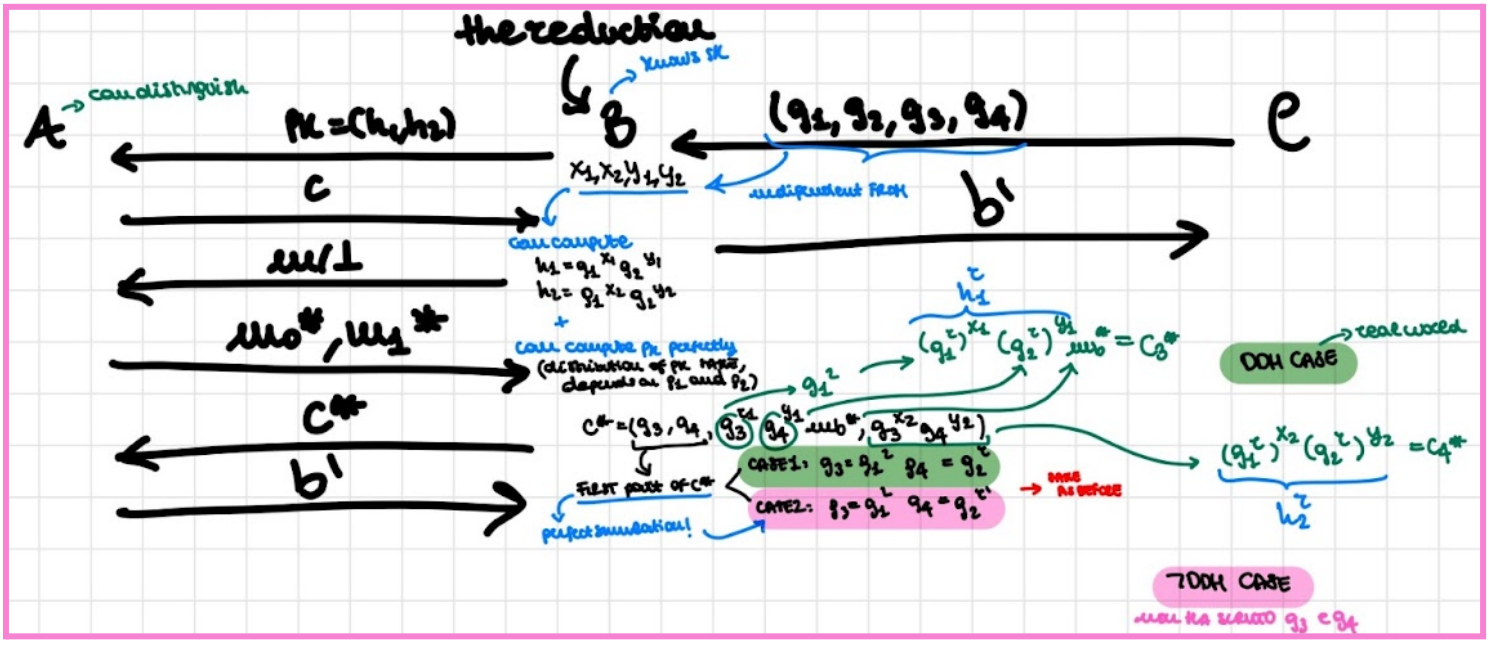
Then we define the hybrid games $H_0(\lambda, b) = Game_{\Pi,A}^{CCA1}(\lambda, b)$ and $H_1(\lambda, b)$, where (differently from H_0) we compute c^* with two different "r", $r, r' \leftarrow \$\mathbb{Z}_q$, in the following way $c^* = (g_1^r, g_2^{r'}, (g_1^r)^{x_1}(g_2^{r'})^{y_1} * m_b^*, (g_1^r)^{x_2}(g_2^{r'})^{y_2})$.

Now, we must prove that $H_0(\lambda, b) \approx_c H_1(\lambda, b), \forall b$ and $H_1(\lambda, 0) \approx_c H_1(\lambda, 1)$.



[Lemma 1] We prove that $H_0(\lambda, b) \approx_c H_1(\lambda, b), \forall b$ by reduction to DDH. Main observation: consider a tuple (g_1, g_2, g_3, g_4)

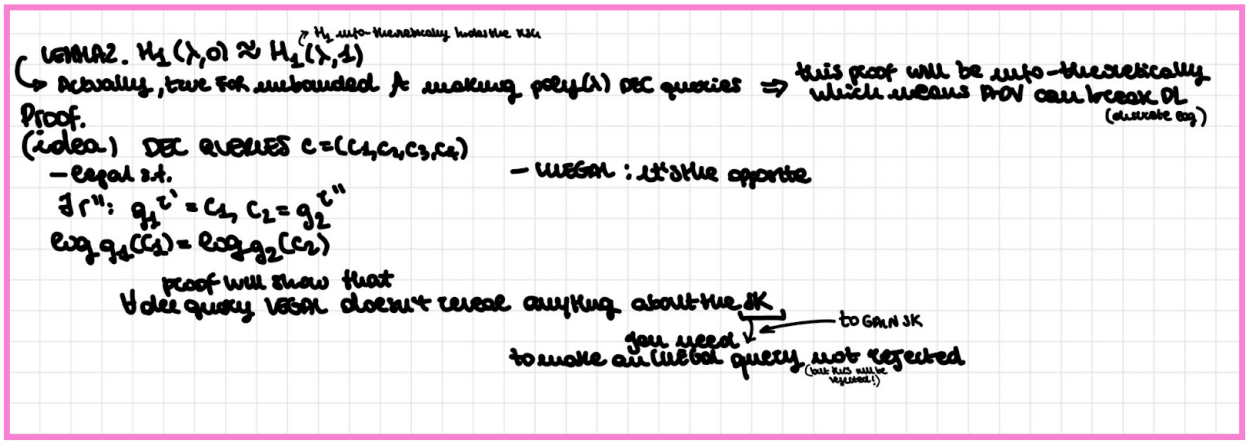
1. in DDH case, $g_1 = g;\ g_2 = g_1^{\alpha};\ g_3 = g_1^{r};\ g_4 = g_1^{\alpha r} = g_2^{r}$ with $\alpha, r \leftarrow \$\mathbb{Z}_q$
2. in not DDH case, $g_1 = g;\ g_2 = g_1^{\alpha};\ g_3 = g_1^{r};\ g_4 = g_1^{r''} = g_2^{r'}$ for some $r', r'' \leftarrow \$\mathbb{Z}_q$ with $\alpha, r \leftarrow \$\mathbb{Z}_q$



The reduction is perfect:

$c_3^* = (g_1^r)^{x_1}(g_2^r)^{y_1} * m_b^* = h_1^* * m_b^*$ and $c_4^* = (g_1^r)^{x_2}(g_2^r)^{y_2} = h_2^*$ → Perfect simulation!

If the adversary can distinguish the two hybrids, the reduction breaks DDH.



[Lemma 2] Lemma 2. $H_1(\lambda, 0) \approx_c H_1(\lambda, 1)$

Now we prove that $H_1(\lambda, 0) \approx_c H_1(\lambda, 1)$. Actually A is unbounded but with $p = poly(\lambda)$ decryption queries. Let $(g_1, g_2 = g_1^{\alpha}, g_3 = g_1^{r}, g_4 = g_2^{r'})$ without loss of generality (wlog) $\alpha \neq 0, r \neq r'$. From pk, $log_{g_1} h_1 = x_1 + (log_{g_1} g_2) y_1 = x_1 + \alpha y_1$ (*).

$\rightarrow q$ possible (x_1, y_1) pairs.

Decryption queries $c = (c_1, c_2, c_3, c_4)$ is legal if exists r'' such that $c_1 = g_1^{r''}, c_2 = g_2^{r''}$ (Note. Legal if the first q elements are the same of discrete log).

[CLAIM 1] A obtains additional information about (x_1, y_1) only if it makes decryption queries that is:

- illegal
- not rejected

Let's prove this. If $Dec(sk, c) = \perp$, A knows $c_4 \neq c_1^{x_2} c_2^{y_2}$ (which reveals nothing about (x_1, y_1)). Assume $Dec(sk, c) \neq \perp$ and c legal, then A knows $m = \frac{c_3}{c_1^{x_1} c_2^{y_1}}$ and so

$log_{g_1} m = log_{g_1} c_3 - x_1 log_{g_1} c_1 - y_1 log_{g_1} c_2 = log_{g_1} c_3 - x_1 r'' - y_1 r''$ (**).

However (*) and (**) are dependent!

$$\begin{pmatrix} 1 & \alpha \\ -r'' & -\alpha r'' \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} \log_{g_1} h_1 \\ \log_{g_1} m - \log_{g_1} c_3 \end{pmatrix}, \quad \det \begin{pmatrix} 1 & \alpha \\ -r'' & -\alpha r'' \end{pmatrix} = 0 \quad \blacksquare$$

[CLAIM 2] Probability of A making decryption query that is illegal and not rejected is negligible. Let's prove this.

$log_{g_1} c_1 = r_1 \neq r_2 = log_{g_2} c_2$.

For (c_1, c_2, c_3, c_4) not rejected it requires $c_4 = c_1^{x_2} c_2^{y_2}$. What does A knows about (x_2, y_2)?

From pk, $log_{g_1} h_2 = x_2 + (log_{g_1} g_2) y_2 = x_2 + \alpha y_2$ (*').

Fix $c_4 \in \mathbb{G}$, for c not rejected $log_{g_1} c_4 = x_2 log_{g_1} c_1 + y_2 log_{g_1} c_2 = x_2 r_1 + y_2 \alpha r_2$ (**').

(*') and (**') are independent.

$$\det \begin{pmatrix} 1 & \alpha \\ r_1 & r_2\alpha \end{pmatrix} = r_2\alpha - r_1\alpha = \alpha(r_2 - r_1) \neq 0$$

→ because (*') and (**') are independent, every value of c_4 is equally likely because for any $c_4 \in \mathbb{G}$ there is unique (x_2, y_2)

→ A can predict c_4 with probability $1/q$.

But if query rejected, it knows c_4 not good. Overall: at query $i \in [p]$ can guess c_4 with probability $1/(q - i)$

→ $Pr[\exists i : (i+1)th\ query\ not\ rejected] \leq \Sigma_{i=0}^{p-1} Pr[(i+1)th\ query\ not\ rejected]$

$\leq p * \frac{1}{q-p} = negl(\lambda)$

→ By the claims except with negligible probability all A knows about (x_1, y_1) before challenge is $log_{g_1} h_1 = x_1 + \alpha y_1$ (*).

Now look at $g_3^{x_1} g_4^{y_1} = h \in \mathbb{G}$ for any h, $log_{g_1} h = x_1 log_{g_1} g_3 + y_1 log_{g_1} g_4 = x_1 r + y_1 \alpha r'$ (***)

$$\det \begin{pmatrix} 1 & \alpha \\ r & \alpha r' \end{pmatrix} = \alpha(r' - r) \neq 0. \quad (wlog)$$

→ Because h is any element, all values of h are equally likely. So h is uniform and m_b^* information theoretically hidden.

- Thm F - CSlite is CCA1 secure under DDH + Lemma 1 (+ proof) + Lemma 2 (+ proof + CLAIM 1 with proof + CLAIM 2 with proof)

Why not CCA2? Because the proof breaks (first lemma breaks).

Given c_4^*, A knows $log_{g_1} c_4^* = x_2 log_{g_1} c_1 + y_2 log_{g_1} c_2$ which is linearly independent of (*')

→ can compute (x_2, y_2).

Exercise. Show CSlite is not CCA2 secure.

This is the Full CS:

- $params = (\mathbb{G}, g_1, g_2, q)$

  $H: \{0,1\}^* \to \mathbb{Z}_q$

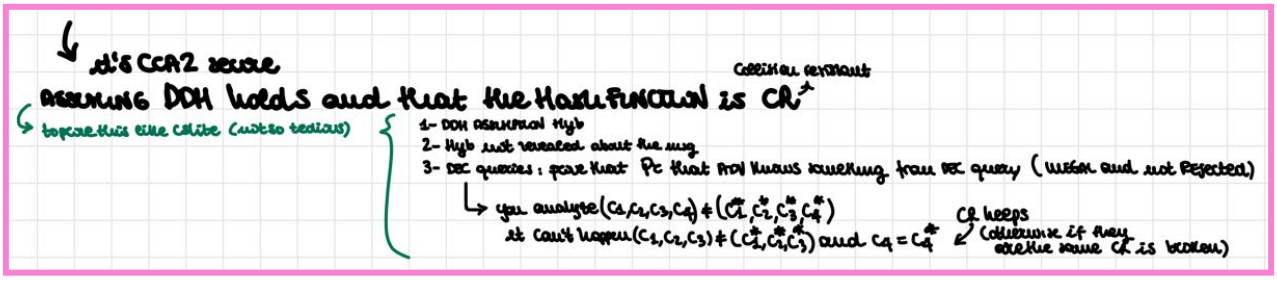
  $sk = (x_1, y_1, x_2, y_2, x_3, y_3) \in \mathbb{Z}_q^6$

  $pk = (params, h_1 = g_1^{x_1} g_2^{y_1}, h_2 = g_1^{x_2} g_2^{y_2}, h_3 = g_1^{x_3} g_2^{y_3})$
- $Enc(pk, m)$: $c = (c_1, c_2, c_3, c_4) = (g_1^r, g_2^r, h_1^r * m, (h_2 * h_3^{\beta})^r)$ with $\beta = H(c_1, c_2, c_3)$

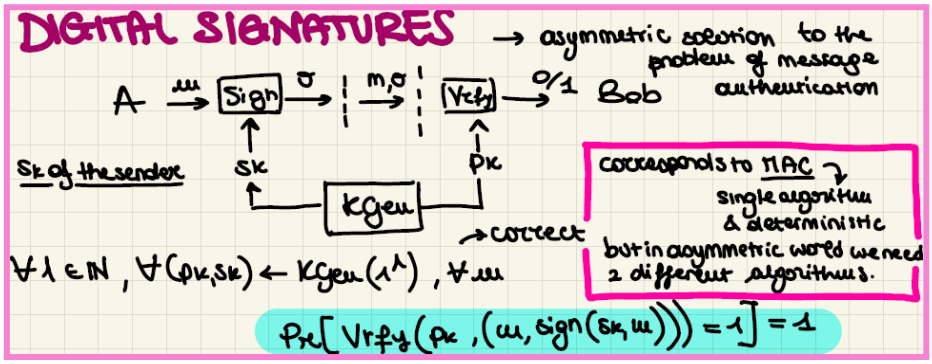
- $Dec(sk, c)$:
  if $c_1^{x_2+\beta x_3} * c_2^{y_2+\beta y_3} = c_4$ with $\beta = H(c_1, c_2, c_3)$

  Output $\dfrac{c_3}{c_1^{x_1} * c_2^{y_1}} = m$

  else Output $\perp$

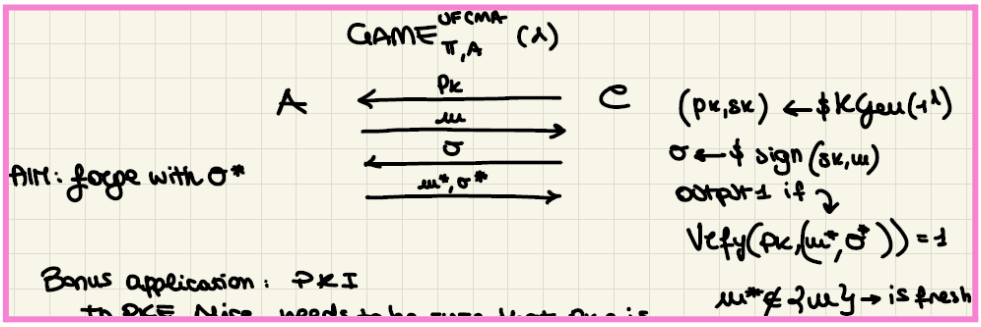


Digital signature



Correctness: $\forall\lambda\in\mathbb{N}, \forall(pk, sk)\in KGen(1^{\lambda}), \forall m,\ Pr[Vrfy(pk, (m, Sign(sk, m))) = 1] = 1$

Signature scheme is a cryptographic primitive, $\Pi = (KGen, Sign, Vrfy)$. What is a secure signature? Π is a secure signature (so Π is UFCMA) if $Pr[Game_{\Pi,A}^{UFCMA}(\lambda) = 1]$ is negligible for each PPT A [Def D - Secure signature]. Nota. (m, σ) a poly number of queries.

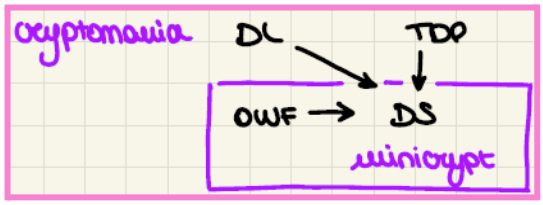


- Def D - Secure signature

Digital Signature are in minicrypt.

[1]: 8 (Tecniche asimmetriche di integrità)

Nel Capitolo 7 abbiamo studiato le tecniche simmetriche per soddisfare il requisito di autenticazione di messaggio, ovvero di

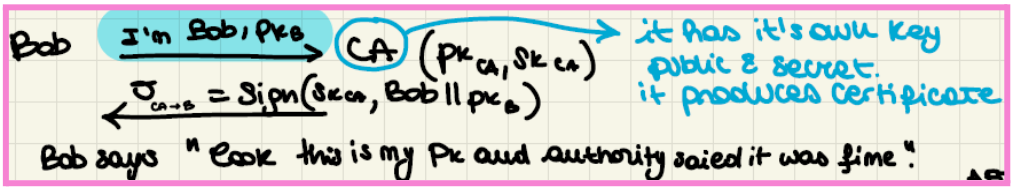


integrità. In questo capitolo ci occupiamo delle firme digitali, che sono l'equivalente dei codici autenticatori di messaggio nel contesto della crittografia asimmetrica. (...)
Come abbiamo già discusso nel Capitolo 6, uno schema crittografico asimmetrico prevede che Alice ed il Bianconiglio posseggano due chiavi: una chiave pubblica ed una chiave segreta. Supponiamo che Alice voglia inviare al Bianconiglio il messaggio m attraverso un canale insicuro, preoccupandosi dell'integrità relativa al messaggio m trasmesso. Esistono primitive che consentano di conservare l'integrità del messaggio m inviato (in chiaro) sul canale, senza condividere a priori un segreto?
L'idea è che Alice produca una firma digitale da associare al messaggio m, in modo che il Bianconiglio possa verificarne l'integrità. Intuitivamente, la firma deve essere generata attraverso la chiave segreta di Alice, in modo che nessun altro possa farlo al suo posto. D'altra parte, la verifica della firma che accompagna il messaggio m deve avvenire tramite la chiave pubblica di Alice, in modo che possa essere effettuata da tutti.
È bene osservare due differenze fondamentali tra il contesto a chiave simmetrica e quello a chiave asimmetrica:

- Pubblica verificabilità. A differenza di un MAC, una firma digitale è verificabile pubblicamente. Supponiamo di essere nel contesto a chiave segreta e che il Bianconiglio abbia ricevuto una coppia valida (m, φ) da Alice. Se ora il Bianconiglio volesse inoltrare l'autenticatore al Cappellaio Matto, quest'ultimo non potrà essere in grado di verificarlo, a meno che non conosca la chiave segreta k condivisa da Alice e dal Bianconiglio. Le firme digitali sono invece trasferibili: il Cappellaio Matto ha bisogno solo della chiave pubblica di Alice per poter verificare una sua firma.
- Non ripudio. Supponiamo che Alice abbia firmato un messaggio m, ma ad un certo punto neghi di averlo fatto. Alcune firme digitali consentono di verificare se una data firma è stata veramente prodotta da Alice, così che quest'ultima non possa negare di averla prodotta.

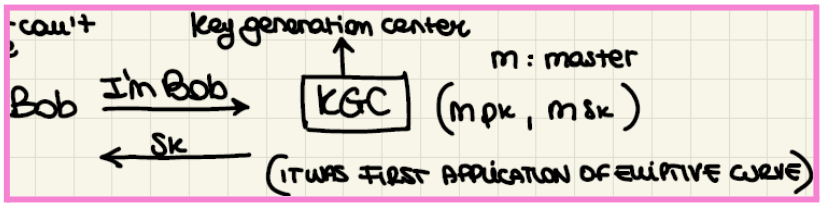
Public key infrastructure and Identity Based Encryption

In PKE, Alice needs to be sure that pk_b is the public key of Bob. Solution: have pk_b "certified" by authority. How? The authority signs the binding between pk and Bob. How can Alice verify the certificate? Alice can verify the certificate with the public key of CA.



(•) How can I trust CA? It's a problem of trust. In real life, there are multiple CA and CA hierarchy (exist a path of certificates that follows CA hierarchy). A certificate has many properties and implementation details; this is the real bottleneck (checking certificates). Can we do better than this (and remove some limitations)? Yes, Identity Based Encryption (IBE).

Identity Based Encryption (IBE) is an idea by Shamir (1984). Starting from the bottleneck that there are these public keys which need to be certified, the idea is → What if the public key doesn't need to be certified? Why do we need to certify them?



What if pk_b is some unique identity of Bob? Bob sends a social security number (or

email) for Bob's key: some string which is uniquely identifier of Bob. This application is the first one of the elliptic curve. Actually nobody knows how to do it from DDH or CDH.

Exercise. IBE⇒CCA security, prove it.

(•) Bob has an identity ID (a string) and Alice knows the identity of Bob. Bob is the only one that can decrypt. But if Bob could generate the secret key by itself from the ID, everyone could do it (it's public). We need authority anyway! It is better than PKE because there are no certificates, but you still need authority in order for it to be secure. Main advantage: no online server needed to encrypt. But it is still not the perfect solution.

[1]: 10 (Crittografia su base identità)

Abbiamo studiato due insiemi di tecniche per la progettazione di sistemi sicuri: tecniche simmetriche e tecniche asimmetriche. Nel contesto delle tecniche simmetriche si assume che le parti coinvolte nel sistema condividano una chiave segreta; pertanto il problema principale è quello di distribuire le chiavi in modo sicuro. (Ci occuperemo di questo, in parte, nel Paragrafo 11.1.) L'uso di tecniche asimmetriche evita a priori questa problematica: Alice possiede una coppia di chiavi (una pubblica e l'altra privata) e se il Bianconiglio vuole inviarle un messaggio, necessita solamente della chiave pubblica di Alice. Sorge un problema di "fiducia": Come può il Bianconiglio essere sicuro che la chiave pubblica di Alice sia "autentica", ovvero corrisponda effettivamente alla chiave segreta di Alice?
Rispondere a questa domanda è fondamentale, altrimenti la Regina Rossa potrebbe semplicemente scambiare la sua chiave pubblica con la chiave pubblica di Alice, così da poter, ad esempio, decifrare tutti i messaggi indirizzati a questa. In questo capitolo studieremo essenzialmente due soluzioni al dilemma dell'autenticità delle chiavi pubbliche.

10.1 (Infrastrutture a chiave pubblica)

La prima soluzione al problema dell'autenticità delle chiavi pubbliche risiede nel concetto di infrastruttura a chiave pubblica (Public Key Infrastructure, PKI). L'idea di base consiste nell'istituzione di una terza parte fidata detta autorità di certificazione (Certification Authority, CA) che si occupi di certificare l'autenticità delle chiavi pubbliche degli utenti. Purtroppo non è possibile contare su una singola CA, essenzialmente perché è impossibile pensare che esista un'entità di cui tutti si fidano indipendentemente dal contesto (e che sia utilizzabile in ogni contesto). Inoltre, una soluzione con una singola CA sarebbe comunque poco scalabile. È bene tener presente che la messa in campo di una PKI, in generale, dipende dalla specifica applicazione crittografica e dal contesto in cui essa è utilizzata.

10.2 (Un'alternativa alle PKI)

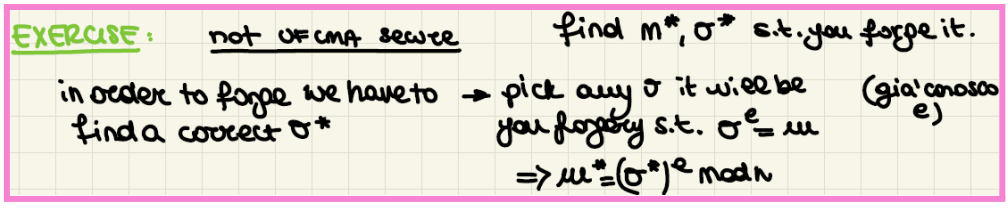
L'idea principale è quella di semplificare il processo di autenticazione delle chiavi pubbliche facendo in modo che la chiave pubblica di Alice sia direttamente la sua "identità digitale". A seconda delle applicazioni tale "identità digitale" può assumere la forma di un indirizzo di posta elettronica oppure di un numero di telefono. In questo modo la chiave pubblica di Alice non necessita di essere preventivamente autenticata. Di conseguenza non sono necessari certificati, la cui gestione (come abbiamo visto) è l'aspetto più complesso nelle PKI. È bene sottolineare sin da ora che la realizzazione concreta del collegamento tra utente ed identità digitale è tutt'altro che banale. Siccome la chiave pubblica di Alice non è ricavata dalla chiave segreta, ma è rappresentata di fatto dalla sua identità, segue che, al contrario di come succede nei sistemi basati sulle PKI, la chiave segreta di Alice deve essere derivata dalla sua identità. Non possiamo però pensare che Alice determini la sua chiave segreta da sola, altrimenti la Regina sarebbe chiaramente in grado di fare lo stesso! Per questo motivo si introduce una terza parte detta centro di generazione delle chiavi (Key Generation Center, KGC) il cui scopo è esattamente quello di generare la chiave segreta di un utente. Per fare ciò il KGC usa una chiave segreta principale msk ed alcuni parametri pubblici $ppub$ (mpk) noti a tutti gli utenti del sistema. Il risultato è la chiave segreta sk_u relativa all'utente con identità u; questi a sua volta potrà usare la chiave segreta per le normali operazioni previste dal crittosistema.

Signing with RSA

What about signing with RSA? →

- $KGen(1^{\lambda})$: (p_k, s_k), $pk = (n, e)$; $sk = (n, d)$
- $Sign(sk, m) = m^d \bmod n$
- $Vrfy(pk, (m, \sigma)) = 1 \Leftrightarrow \sigma^e = m \bmod n$

Exercise (solution in the book, + idea). This scheme is not UFCMA secure. Fix $\sigma^* \epsilon \mathbb{Z}_n^*$ and output $m^* = (\sigma^*)^e \bmod n$



[1]: 8.2 (Naïve RSA e hash a dominio pieno)

Come primo esempio di schema di firma digitale useremo il crittosistema RSA in modo diretto. (...) L'idea di base è quella di scambiare il ruolo degli elementi d ed e.

La correttezza è assicurata dal piccolo Teorema di Fermat e dal fatto che $e * d \equiv 1 \, (mod \, \varphi(N))$, ovvero $e * d \equiv 1 + r * \varphi(N)$ per qualche $r \epsilon \mathbb{N}$. Possiamo infatti scrivere: $\sigma^e \equiv m^{e*d} \equiv m^{1+r*\varphi(N)} \equiv m \cdot m^{r*\varphi(N)} \equiv m * m \, (mod \, N)$.

Naïve RSA è insicuro. Mostreremo ora che l'uso naïve di RSA è totalmente insicuro. Il primo attacco che descriviamo è basato sulla sola conoscenza della chiave pubblica $pk = (N, e)$. L'attaccante sceglie $\sigma^* \leftarrow \$ \mathbb{Z}_n^*$ e calcola $m^* = (\sigma^*)^e \, mod \, N$: la coppia (m^*, σ^*) è ovviamente valida. Sebbene l'avversario non abbia alcun controllo sul risultato (si parla infatti di forgiatura esistenziale), la Definizione 8.2 non è rispettata.

Possiamo produrre anche una forgiatura selettiva (ovvero una forgiatura in cui l'avversario ha pieno controllo sul messaggio m^* di cui forgia la firma). Supponiamo che A voglia produrre la firma relativa al messaggio $m \leftarrow \$ \mathbb{Z}_n^*$. L'attaccante può scegliere $m_1 \leftarrow \$ \mathbb{Z}_n^*$, porre $m_2 = m/m_1 \, mod \, N$ e chiedere all'oracolo $Sign_{sk}(\cdot)$ le firme σ_1, σ_2 relative ad m_1, m_2. Allora $\sigma = \sigma_1 * \sigma_2$ è una firma valida per m. Infatti $\sigma^e \equiv (\sigma_1 * \sigma_2)^e \equiv (m_1 * m_2)^{ed} \equiv m^{ed} \equiv m \, (mod \, N)$.

Hash a dominio pieno. Un modo per evitare questi attacchi è quello di usare una funzione hash. Come vedremo la dimostrazione di sicurezza non è nel modello standard, ma nel modello dell'oracolo casuale. L'idea di base è quella di calcolare l'hash del messaggio prima di produrre la firma. Tra l'altro questo approccio ha il vantaggio che possiamo firmare messaggi a lunghezza arbitraria, in quanto l'hash comprime il messaggio iniziale in poche centinaia di bit (codificabili poi in $\mathbb{Z}_n^*$). Lo schema, detto hash a dominio pieno (Full Domain Hash, FDH), è presentato nel Crittosistema 8.2.

Exercise (+ idea). Show that exists an adversary that can forge any message in poly time.

Do it with FDH (Full Domain Hash) $\sigma^e = H(m)^d mod n$.

Lecture 21

[1]: 4, 8, 13 [2]: 5, 12 [3]: 6, 7 [7]: 6.2

- Signature schemes.
- The random oracle model (ROM).
- Constructing signatures from TPDs (Full-Domain Hash).
- Identification (ID) schemes.
- Passive security and canonical ID schemes.
- The Fiat-Shamir transform.

FDH (Full Domain Hash)

It is a signature from RSA (TDP, $(KGen, f, f^{-1})$) [Note: $H: \{0,1\}^* \to \mathfrak{X}_{pk}$] →

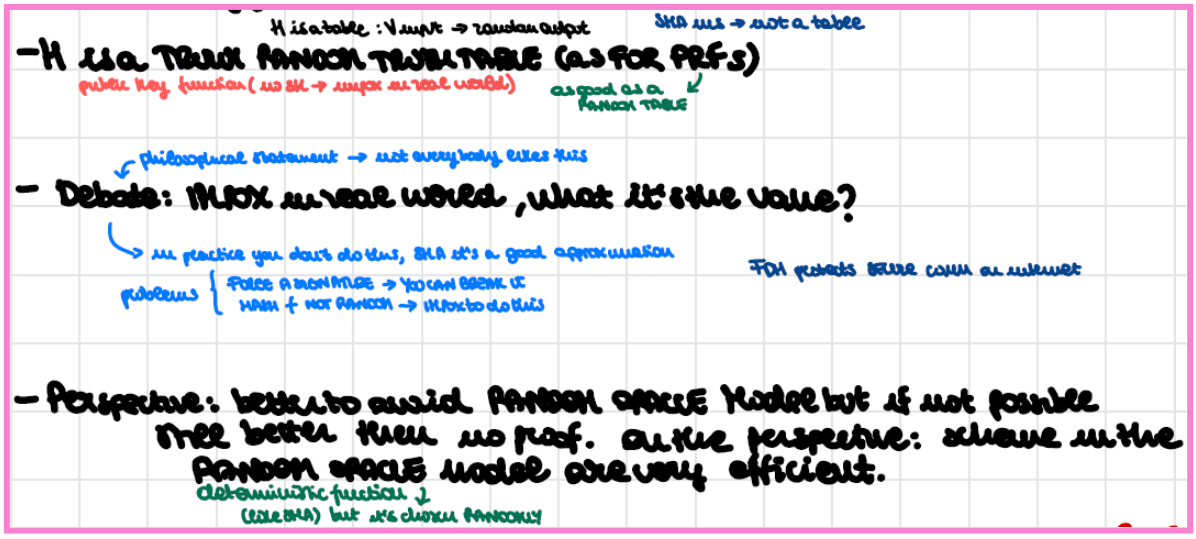
- $KGen(1^\lambda)$: (p_k, s_k) (RSA, $p_k = (n, e)$, $s_k = (n, d)$)
- $Sign(s_k, m)$: $f_{sk}^{-1}(H(m)) = \sigma$, we hash the message and we invert it

  (RSA, $\sigma = (H(m))^d \bmod n$)
- $Vrfy(p_k, m, \sigma)$: $H(m) = f_{pk}(\sigma)$

  (RSA, $H(m) = \sigma^e \bmod n$)

We will prove that this scheme (FDH) is UF-CMA (Thm G), but first we need to talk about the random oracle methodology (ROM).

Considerations about ROM.



[1]: 4.4 (Modello dell'oracolo casuale)

Vedremo diverse applicazioni delle funzioni hash nei capitoli successivi, ad esempio nel contesto della confidenzialità e dell'integrità di messaggio. Come sappiamo l'approccio della sicurezza computazionale basa la sicurezza delle primitive crittografiche sull'ipotesi che alcuni problemi computazionali siano intrattabili date le risorse a disposizione di un attaccante. Purtroppo a volte da sola questa ipotesi non è sufficiente (o comunque conduce a costruzioni inefficienti).
Per offrire un piano intermedio tra sicurezza formale ed efficienza si è soliti introdurre un modello idealizzato, in cui le primitive crittografiche possono essere dimostrate sicure. L'esempio più famoso di tale metodologia è il modello dell'oracolo casuale. In questo modello si suppone l'esistenza di un

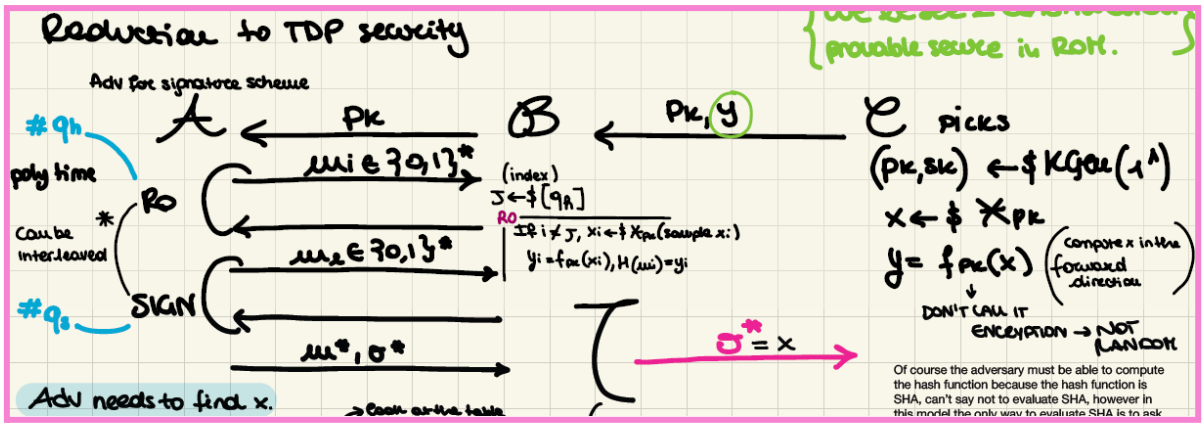
oracolo casuale $H(\bullet)$ interrogabile da tutti: dato un input x l'oracolo restituisce il valore $H(\bullet)$ e l'unico modo per calcolare $H(\bullet)$ è inviare x all'oracolo. Si è soliti parlare di modello standard quando non è necessario utilizzare un oracolo casuale e quindi una primitiva può essere dimostrata sicura senza tale ipotesi. Più precisamente nel modello dell'oracolo casuale si fanno le seguenti ipotesi:

1. Si assume che le richieste all'oracolo siano locali, nel senso che se qualcuno interroga l'oracolo con input x, la stringa x resta segreta e nessuno è in grado di determinare nemmeno che la richiesta ha avuto luogo.
2. L'oracolo è consistente, ovvero se l'oracolo restituisce y in corrispondenza della stringa x, allora restituirà sempre e comunque y quando viene interrogato per x.
3. La funzione $H(\bullet)$ è scelta a caso (una volta per tutte) tra tutte quelle possibili. Supponiamo che $H(\bullet)$ mappi n bit in $l(n)$ bit. Una funzione di questo tipo è rappresentabile con una stringa lunga $2^n l(n)$ bit e quindi ci sono $2^{2^n l(n)}$ possibili mappe. Scegliere $H(\bullet)$ a caso significa scegliere una di queste mappe a caso tra tutte le mappe possibili. Notare che, vista la dimensione, è decisamente impossibile fare ciò in pratica. Equivalentemente, è possibile pensare che la funzione $H(\bullet)$ sia costruita "al volo". Data una richiesta (x_i, y_i) si controlla se la stringa xi è stata già richiesta: in caso affermativo si restituisce la stessa stringa usata in precedenza, altrimenti l'output è scelto a caso in $\{0,1\}^{l(n)}$. Sebbene ciò sia del tutto indifferente dal punto di vista di chi interroga l'oracolo, costruire $H(\bullet)$ "al volo" è sensibilmente diverso da supporre che $H(\bullet)$ sia disponibile una volta per tutte quando una primitiva è inizializzata.
4. In una prova per riduzione è possibile "programmare" l'oracolo opportunamente, in modo che i valori restituiti siano utilizzabili in modo conveniente (purché essi appaiano uniformemente casuali).

Critiche e punti a favore. Nella realtà, in fase d'implementazione, dobbiamo sostituire l'oracolo con una funzione concreta $\hat{H}$, ad esempio SHA-1. Da qui la domanda: qual è il valore di una dimostrazione di sicurezza nel modello dell'oracolo casuale quando tale schema è utilizzato nel mondo reale? La risposta è per alcuni versi controversa.
Un celebre risultato negativo mostra che esistono primitive crittografiche che sono sicuri nel modello dell'oracolo casuale, ma che sono insicuri per ogni possibile implementazione concreta dell'oracolo stesso! Tuttavia, è importante sottolineare che questi schemi sono completamente "artificiali" e non esiste nessun attacco concreto ad un sistema sicuro nel modello dell'oracolo utilizzato in pratica.
Il punto è che non è affatto chiaro cosa significhi in pratica per una funzione reale emulare un oracolo casuale. Ad esempio un oracolo casuale è completamente diverso da una funzione pseudocasuale (PRF): mentre la seconda è una funzione con chiave che può essere valutata solo una volta che la chiave è nota e sembra puramente casuale altrimenti, il primo è una funzione senza chiave che chiunque può valutare e che nonostante ciò deve sembrare puramente casuale in un senso che non è ben definibile.
Per tutti questi motivi, una dimostrazione nel modello standard è considerata in genere più valida di una dimostrazione nel modello dell'oracolo casuale. Tuttavia, gli schemi con sicurezza dimostrabile nel modello dell'oracolo casuale sono spesso molto più efficienti; inoltre si tende a pensare che una dimostrazione nel modello dell'oracolo è sempre meglio che non avere alcuna dimostrazione. Una prova in questo modello idealizzato, infatti, dà un senso di validità allo schema, sottolineando che le uniche sue debolezze sono quelle che possono sorgere rimpiazzando l'oracolo con una funzione concreta. Quindi, se abbiamo modo di ritenere che tale funzione è "sufficientemente buona", possiamo confidare in un certo senso che lo schema sia sicuro. Dal punto di vista teorico sarebbe auspicabile formalizzare il requisito di essere "sufficientemente buona". In conclusione usare uno schema che ha una prova formale nel modello dell'oracolo è decisamente meglio che usarne uno che non ha alcuna dimostrazione di sicurezza! Tuttavia se esiste uno schema equivalente sicuro nel modello standard quest'ultimo è preferibile (almeno a parità di efficienza).

We define a theorem [Thm G - The above scheme (FDH) is UF-CMA, assuming TDP is one way and H modeled as a random oracle]. To prove this, we will use a technique heavily based on ROM.
The proof is based on a reduction to TDP security: we assume by contradiction…

- $\exists A^{FDH}$ that breaks FDH (it is UF-CMA) with probability no negligible and
- $\exists B^{TDP}$ that breaks TDP with probability no negligible.



Of course the adversary must be able to compute the hash function because the hash function is SHA, can't say not to evaluate SHA, however in this model the only way to evaluate SHA is to ask the oracle and the only way to compute the function is to query, to look at the table. Consequences: whenever the adversary want to evaluate the hash function must ask the oracle, if it asks the Oracle explicitly the reduction is gonna
see it and in particular the reduction must simulate these queries. The adv want to see the hash of m_i, the reduction must answer otherwise this guy disappears and doesn't break the scheme and the TDP.
Sequence →
C computes $(pk, sk) \leftarrow \$KGen(1^{\lambda})$, $x \rightarrow \$\mathfrak{X}_{pk}$ (permutation of x) and $y_{pk}(x)$ (don't call it encryption, it is not random, i.e. not CPA secure; it computes s in the forward direction). Then sends pk and y to B^{TDP}.
B^{TDP} sends pk to A^{FDH}.
RO queries (these are done $\$q_h$ times):

- A^{FDH} sends $m_i \in \{0,1\}^*$ to B^{TDP}
- B^{TDP} picks a random index $j \leftarrow \$[q_h]$.

  RO:

  if $i \neq j$, then $x_i \leftarrow \$\mathfrak{X}_{pk}$ $y_i = f_{pk}(x_i)$, $H(m_i) = y_i$

  else $H(m_j) = y$
- B^{TDP} sends y_i to

$Sign$ queries (these are done $\$q_s$ times):

- A^{FDH} sends $m_i \in \{0,1\}^*$ to B^{TDP}
- $Sign$: upon m_i return $x_i = \sigma_i$

  x_i is the inverse of permutation
- B^{TDP} sends $x_i = \sigma_i$ to A^{FDH}

A^{FDH} sends (m^*, σ^*) to B^{TDP}.

FORGE: (m^*, σ^*)

if $m^* \neq m_j$, then abort

else σ^*

B^{TDP} sends σ^* (that is x, the pre-image of y) to C.

Assumptions:

1. Wlog (without loss of generality) A never repeats queries
2. Wlog before asking m_i signature (or m^* forge), A asks m_i (or m^*) to ROM → m_i (m^*) must be queried to ROM if it's queries (or forged).

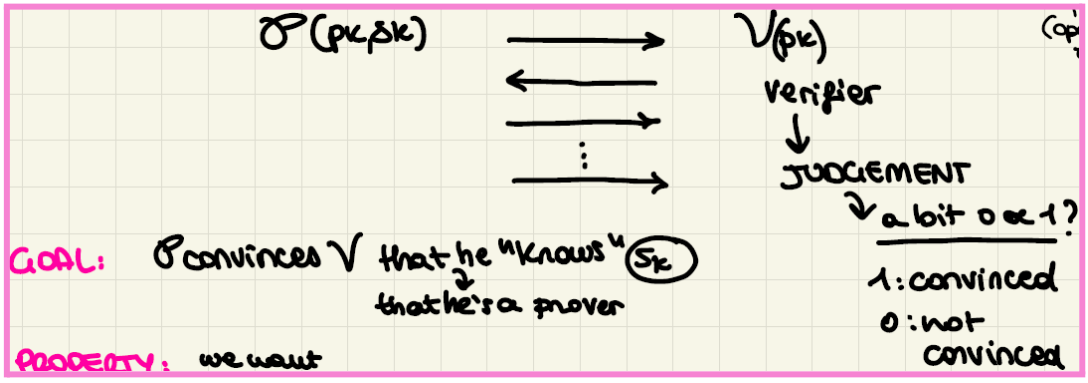
From this analysis, we have that $Sign$ and RO queries are perfectly simulated:

$\sigma^* = x = f_{sk}^{-1}(y)$ with probability $1/poly$ $(Pr[m^* = m_j]Pr[A^{FDH} wins] = (1/\#q_h) * noNegl)$.

- Thm G - The above scheme (FDH) is UF-CMA, assuming TDP is one way and H modeled as a random oracle (+ proof)

Identification (ID) schemes

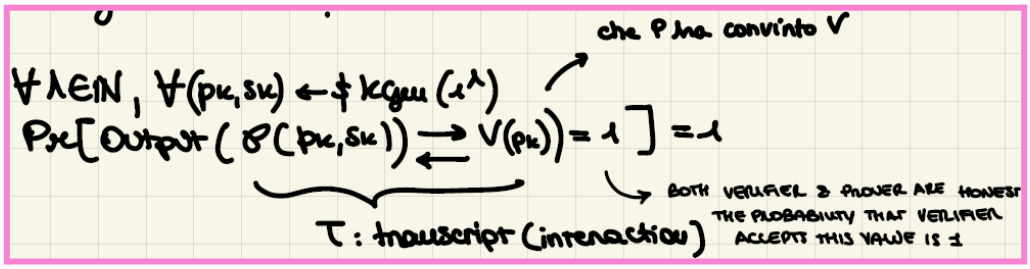
(•) This protocol is used to manages the interaction between Alice and Bob: Alice has to convince Bob that she knows sk and Bob wants to be sure that he is talking to Alice. So Bob has to recognize Alice as the other end of the conversation. We will study a particular class of ID schemes that allows us to obtain digital signatures efficiently.



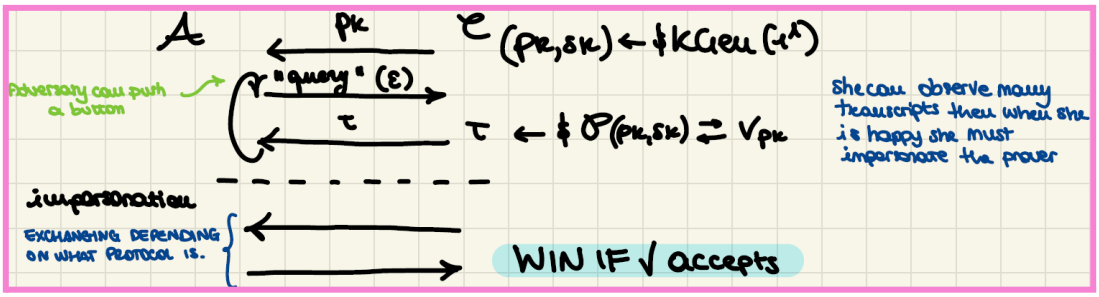
Goal: $\mathcal{P}$ convinces $\mathcal{V}$ that he "knows" s_k, i.e. that he's a prover.

We want these properties…

- Correctness → Honest $\mathcal{P}$ always convinces $\mathcal{V}$?

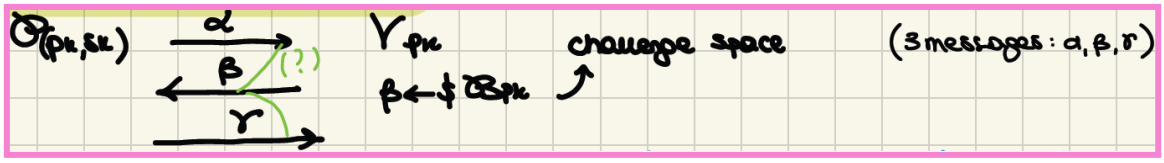


- Secure many definitions dealing with impersonation → For us, passive security which means Eve can't impersonate $\mathcal{P}$, even observing poly many transcripts.
  (•) So the adversary will be passive, i.e. he only observes the communication between honest Alice and honest Bob.



We will be happy with passive security if this cannot be done (no adversary can impersonate).

Canonical ID schemes (Σ-protocol)



It is connected with Zero Knowledge (passive security): I can convince you that I know the secret key even if I don't know it.

From a canonical IDS we want the property of non-degeneracy, i.e. it's hard to predict the first message of the prover (vedi dopo [7]): $\forall \hat{a},\ Pr[\alpha = \hat{a},\ \alpha \leftarrow \$\mathcal{P}(pk;\ sk)] = negl(\lambda)$ (α has high min-entropy).

Example. $\alpha = g^a$ for $a \leftarrow \mathbb{Z}_q$

[7]: 6.2 (Identification Schemes)

An Identification Scheme (IDS) is an interactive protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$.

Definition 31 (Identification Scheme). An IDS is a tuple $\Pi = (KGen, \mathcal{P}, \mathcal{V})$, τ is the message exchange between $\mathcal{P}$ and $\mathcal{V}$, denoted as $\tau \leftarrow \$(\mathcal{P}(pk;\ sk)) \leftrightarrows \mathcal{V}(pk))$, with $(pk;\ sk) \leftarrow \$KGen(1^\lambda)$.

$Output(\mathcal{P}(pk;\ sk)) \leftrightarrows \mathcal{V}(pk))$ is a random variable, and decides if $\mathcal{P}$ knows sk or not.

(...)

Definition 33 (Canonical IDS). In a canonical IDS we have that $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2)$ and that $\mathcal{V} = (\mathcal{V}_1, \mathcal{V}_2)$. Just three messages are exchanged:

1. $\alpha \leftarrow \$\mathcal{P}_1(pk;\ sk)$ is sent to $\mathcal{V}$;
2. $\beta \leftarrow \$\mathcal{B}_{pk} = \mathcal{V}_1$, the challenge, is sent to $\mathcal{P}$;
3. $\gamma \leftarrow \$\mathcal{P}_2(pk,\ sk, \alpha, \beta)$ is sent to $\mathcal{V}$;
4. $\mathcal{V}$ outputs $\mathcal{V}_2(pk, \alpha, \beta, \gamma) \in \{0, 1\}$

The verifier is randomized: this is called a (three-move) "public coins" verifier. Sometimes $\mathcal{P}_1$ and $\mathcal{P}_2$ share a state, i.e., $(\alpha, a) \leftarrow \$\mathcal{P}_1(pk;\ sk)$ and $\gamma \leftarrow \$\mathcal{P}_2(pk,\ sk, \alpha, \beta, a)$.

From a canonical IDS we want the property of non-degeneracy, i.e. it's hard to predict the first message of the prover: $\forall \hat{a},\ Pr[\alpha = \hat{a},\ \alpha \leftarrow \$\mathcal{P}_1(pk;\ sk)] = negl(\lambda)$.

(...)

Construction 24 Fiat-Shamir Transform). Let $\Pi = (KGen, \mathcal{P}, \mathcal{V})$ be an IDS. Consider the signature scheme $\Pi' = (KGen, Sign, Vrfy)$ defined as:

- $KGen(1^\lambda)$: (p_k, s_k)
- $Sign(sk, m)$:

1. $\alpha \leftarrow \$\mathcal{P}_1(pk;\ sk)$;
2. $\beta = H(\alpha, m)$ with $H: \{0,1\}^* \to \mathcal{B}_{pk}$;
3. $\gamma \leftarrow \$\mathcal{P}_2(pk,\ sk, \alpha, \beta)$;
4. $\sigma = (\alpha, \gamma)$;

- $Vrfy(pk, (m, \sigma))$ computes $\beta = H(\alpha, m)$, then returns $\mathcal{V}_2(pk, \alpha, \beta, \gamma)$.

Theorem 42. If Π is passively secure, Π' as defined in Construction 24 is UFCMA.

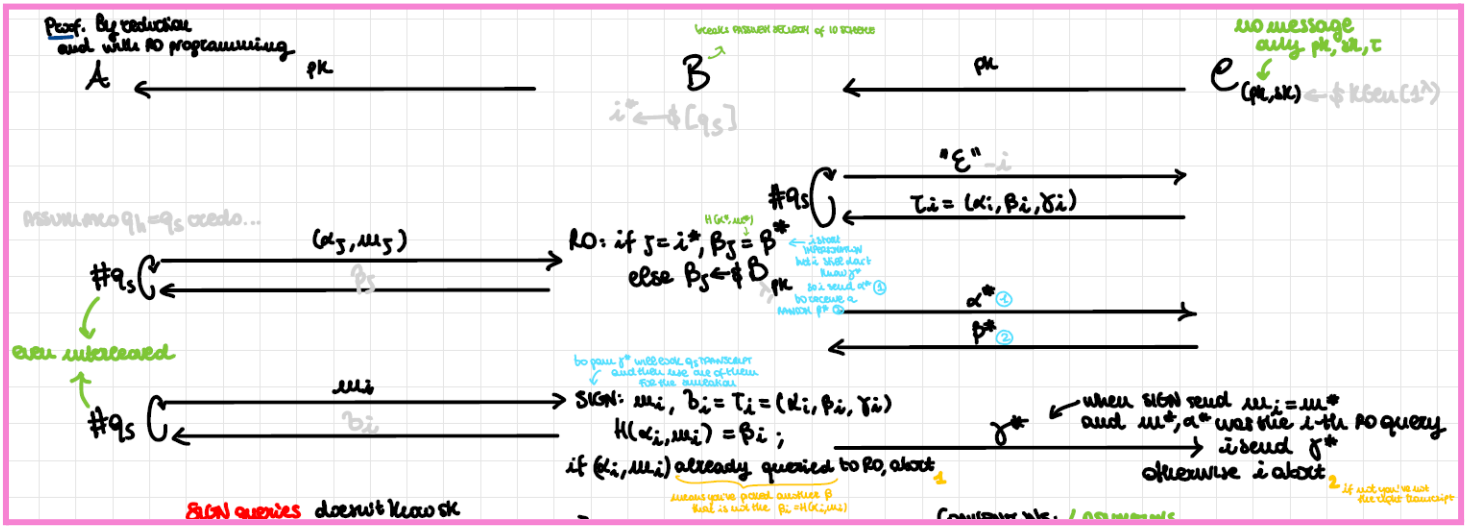
PLAN: 1. FS signatures, 2. Construct ID scheme (next time)

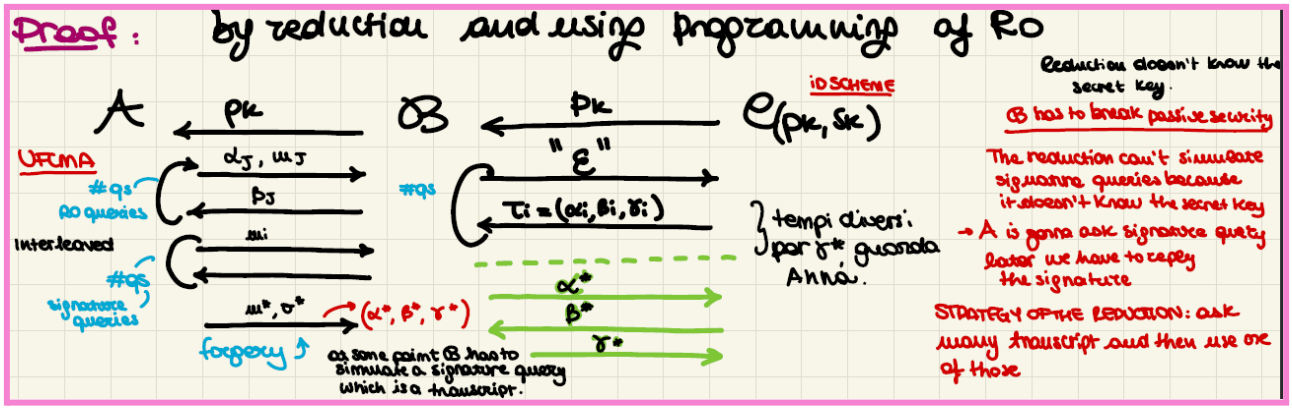
Fiat-Shamir signatures

It is a popular way to derive signatures from ID schemes.
(•) In order to generate a transcript to be accepted, you need the sk. Idea: make the identification scheme non interactive, using hash functions.

- $KGen(1^\lambda)$: (p_k, s_k)
- $Sign(sk, m)$:
  Get α from $\mathcal{P}(pk;\ sk)$
  Let $\beta = H(\alpha, m)$
  Get γ from $\mathcal{P}(pk;\ sk)$
  $\tau = (\alpha, \beta, \gamma)$
- $Vrfy(pk, \sigma)$:
  Let $\beta = H(m, \alpha)$
  Ask $\mathcal{V}(pk, \tau = (\alpha, \beta, \gamma))$

We can prove that the above signature is UF-CMA in the ROM assuming the ID scheme is canonical (non-degenerate) and passively secure [Thm H - FS yields UF-CMA signature in the ROM, assuming ID scheme is passively secure (and non-degenerate)].





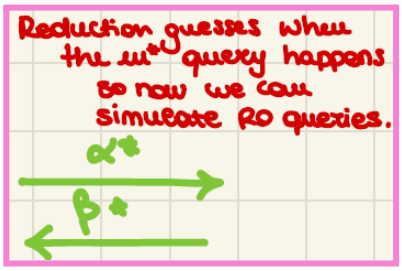
Conventions →

1. A never repeats queries
2. For each signature query m, the adversary knows $\beta = H(m, \alpha)$ (β is the hash of (m, α)), so afterwards doesn't query (m, α) to RO
3. Forge m^* then (m^*, α^*) as RO query

Analysis →

- RO queries (simulated by β_j)

  if $j = i^*$: start the impersonation (because now he knows α^*), send α^*, the reply is β^*. Now reduction makes a break, because it doesn't know γ^* yet. Now $\beta_j = \beta^*$ chosen randomly.



You still can't send γ^*, keep the simulation until you know γ^* and hope it works.

After I do signature query m_i, $\sigma_i = \tau_i = (\alpha_i, \beta_i, \gamma_i) \rightarrow \beta_j = \beta^* = H(m, \alpha)$. I program the random oracle in the sense that if the adversary at some point will ask me the (m_i, α_i) query, I gonna reply with β_i ($\beta_i = H(m_i, \alpha_i)$).

What if the adversary already asked for (m_i, α_i) in RO query? You pick another β different from the β_i before.

- Signature queries on m_i, $G_i = \tau_i = (\alpha_i, \beta_i, \gamma_i)$, $\beta_i = H(m_i, \alpha_i)$

  What if the adversary already asked for (m_i, α_i) → abort

- When A outputs m^*, $(\alpha^*, \beta^*, \gamma^*)$ check that (m^*, α^*) was the i-th RO query

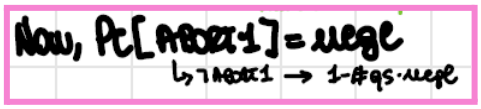
  if yes, then send γ^* to C

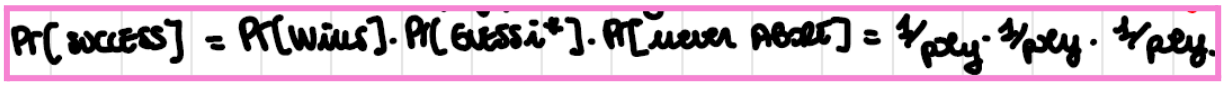
  else abort

Key ideas of simulation →

- $Sign$ queries doesn't know sk, use transcript → program the random oracle in such a way that $\beta_i = H(m_i, \alpha_i)$
- RO queries: try to guess the query where he makes (m^*, α^*). Start impersonation: forges and output γ^*.

Now…





- Thm H - FS yields UF-CMA signature in the ROM, assuming ID scheme is passively secure (and non-degenerate) (+ proof)

Lecture 22

[1] 13.3, 13.4 [2]: 12.5 [3]: 8 [7]: 6.2

- ➢ Definition of Honest Verifier Zero Knowledge (HVZK) and Special Soundness (SS) for canonical ID schemes.
- ➢ Proof that HVZK plus SS imply passive security.
- ➢ Canonical ID schemes from Discrete Log.

Honest Verifier Zero Knowledge (HVZK) and Special Soundness (SS) for canonical ID schemes

[1]: 13 (Conoscenza nulla)

Nei capitoli precedenti ci siamo principalmente occupati di quali requisiti deve soddisfare un protocollo di autenticazione sicuro e di come realizzare tali protocolli. In questo capitolo, ci occuperemo di uno scenario più generale. Immaginiamo che Alice voglia convincere lo Stregatto della veridicità di una certa affermazione. È possibile per Alice convincere lo Stregatto della veridicità dell'affermazione, senza rivelare null'altro se non il fatto che l'affermazione è appunto veritiera? Ciò è in effetti possibile, come vedremo, per ogni "linguaggio" in NP e la risposta a questa domanda risiede nei protocolli a conoscenza nulla (Zero Knowledge, ZK), oggetto del presente capitolo. (...)

Definizioni su base simulazione. Per formalizzare la proprietà di conoscenza nulla, useremo una definizione su base "simulazione". Richiederemo che tutto ciò che la Regina Rossa è in grado di imparare osservando un'interazione tra Alice e lo Stregatto, possa essere perfettamente "simulato" dalla Regina Bianca (detta anche il "simulatore") senza eseguire alcun protocollo. Ciò intuitivamente significa che l'interazione tra Alice e lo Stregatto non ha rilasciato alcuna informazione se non la veridicità o la falsità dell'affermazione stessa.

13.1 Sistemi di prova a conoscenza nulla

In un sistema di prova interattivo, un "dimostratore" $\mathcal{P}$ (i.e., Alice) tenta di convincere un verificatore $\mathcal{V}$ (i.e., lo Stregatto) della veridicità di una certa affermazione. Formalmente, la terminologia "veridicità di una certa affermazione" è espressa attraverso il concetto di relazione.

(...)

Informalmente, un sistema di prova interattivo è uno scambio di messaggi tra $\mathcal{P}$ e $\mathcal{V}$ in cui, fissato un linguaggio L, il dimostratore tenta di convincere il verificatore che un dato elemento x appartiene ad L. Indicheremo tale interazione con $\mathcal{P}(\cdot)\leftrightarrows\mathcal{V}(\cdot)$ e diremo che l'output è accetta quando $\mathcal{V}$ è convinto che $x\epsilon L$ dopo aver interagito con $\mathcal{P}$.

(...)

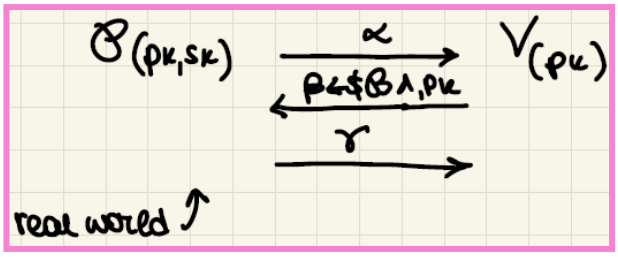
Conoscenza nulla. Intuitivamente, diremo che un sistema di prova interattivo (per un linguaggio L) è ZK se tutto ciò che può essere ricavato dopo aver interagito con $\mathcal{P}$ per un input x, può essere calcolato, a partire da x, senza interagire con $\mathcal{P}$. È importante sottolineare che ciò deve valere per

ogni modo efficiente di interagire con $\mathcal{P}$, non necessariamente quello prescritto da un'esecuzione onesta del protocollo. Formalmente:
Definizione 13.3 (Conoscenza nulla computazionale). (...)
(...)
Conoscenza nulla con verificatore onesto. Un modo per rilassare la Definizione 13.3 è supporre che $\mathcal{V}$ sia sempre onesto nell'esecuzione del protocollo e tenti di imparare qualcosa di non lecito solamente a partire da ciò che ha imparato al termine dell'interazione con $\mathcal{P}$. Ciò modella avversari "onesti ma curiosi" che intercettano un'esecuzione del protocollo tra $\mathcal{P}$ e $\mathcal{V}$.
Definizione 13.4 (Conoscenza nulla con verificatore onesto). (...)

Last time we saw that passively secure ID schemes imply UFCMA signatures.
We can think on it as IP (interactive proof) when $\mathcal{P}$ convince $\mathcal{V}$ it knows sk corresponding to pk in a way that

- SOUND. If A does not know sk, can't convince $\mathcal{V}$
- $(HV)ZK$. Honest proof reveals nothing about sk (if $\mathcal{V}$ semi-honest)

Today we construct ID schemes with passive security and we will use two criteria: HVZK e sound.



[Def E - HVZK (Honest Verifier Zero Knowledge)]
$\Pi = (KGen, \mathcal{P}, \mathcal{V})$ is HVZK if $\exists PPT$ simulator S such that

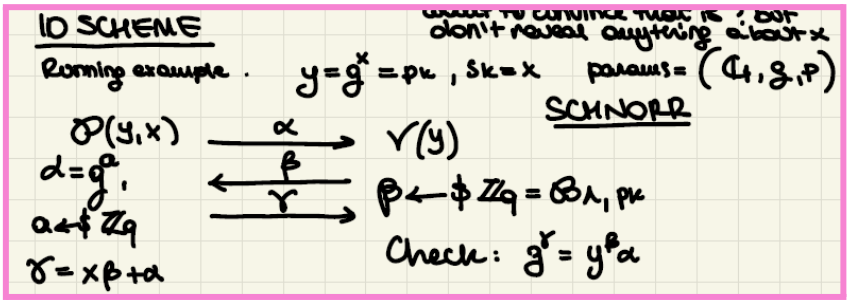
$\{(pk, sk, (\alpha, \beta, \gamma)), (pk, sk) \leftarrow \$KGen(1^{\lambda}); (\alpha, \beta, \gamma) \leftarrow \$\mathcal{P}(pk;\ sk) \leftrightarrows \mathcal{V}(pk)\}$ (real transcript)

$\approx_c \{(pk, sk, (\alpha, \beta, \gamma)), (pk, sk) \leftarrow \$KGen(1^{\lambda}); (\alpha, \beta, \gamma) \leftarrow \$S(pk)\}$ (simulated transcript)

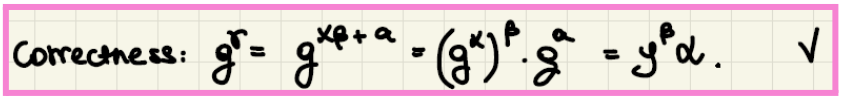
HVZK says you don't learn anything from a transcript. It's called the simulation paradigm: if a protocol is HVZK, then exists a simulator capable of simulating the transcripts ([7]: Definition 34 (Honest Verifier Zero Knowledge)).

- Def E - HVZK (Honest Verifier Zero Knowledge)

Running example: Schnorr protocol, a canonical ID scheme from Discrete Log.



1. Correctness →



2. HVZK →

Consider a simulator $S(y)$

- Pick $\beta, \gamma \leftarrow \mathbb{Z}_q$
- Let $\alpha = g^{\gamma} y^{-\beta}$
- Output(α, β, γ)

Easy to see some distribution of real (α, β, γ) (perfect HVZK).

$L = \{y \exists s \ s.t. \ y = g^x\} \rightarrow$ If y not in the language, prove can't convince me. Fact that sk exists is trivial. If you don't know x, you can't impersonate me.

[Def F - Sound]

$\Pi = (KGen, \mathcal{P}, \mathcal{V})$ is SS (special soundness) if $\exists PPT\ A$

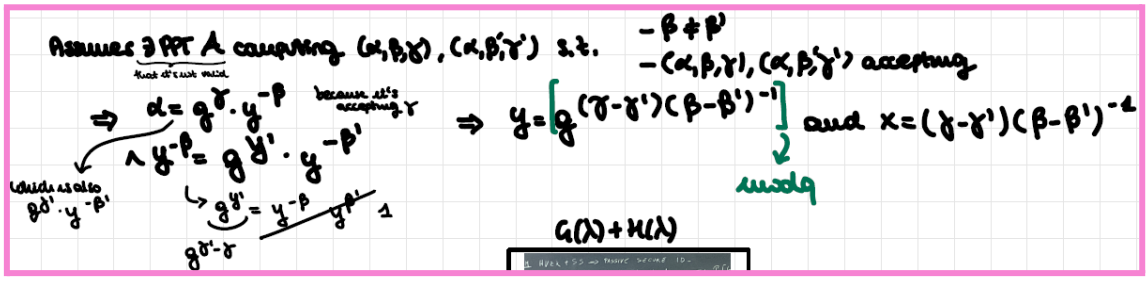
$Pr[\mathcal{V}((\alpha, \beta, \gamma), pk) = \mathcal{V}((\alpha, \beta', \gamma'), pk) = 1 \ with \ \beta \neq \beta':$

$(pk, sk) \leftarrow \$KGen(1^\lambda) \ and \ (\alpha, \beta, \beta', \gamma, \gamma') \leftarrow A(pk)] \leq negl(\lambda)$

This property is about canonical IDS: it's hard to have two transcripts (α, β, γ) and $(\alpha, \beta', \gamma')$ ([7]: Definition 35 (Special Soundness)).

- Def F - Sound

3. Sound →



Assume $\exists PPT\ A(y)$ computing

$\alpha = g^\gamma y^{-\beta} = g^{\gamma'} y^{-\beta'}$

$y^{(\beta'-\beta)} = g^{(\gamma'-\gamma)}$

$g^x = y = g^{(\gamma'-\gamma)(\beta'-\beta)^{-1}}$

$x \equiv (\gamma' - \gamma)(\beta' - \beta)^{-1} \ mod \ q$

→ can break DL

[7]: Construction 25 (Schnorr Protocol)

Construction 25 (Schnorr Protocol). Schnorr protocol is defined as follows:

- KGen(1^λ): $(\mathbb{G}, g, q) \leftarrow$ GroupGen(1^λ), $sk = x \leftarrow \$\mathbb{Z}_q$, $pk = y = g^x$;
- $\mathcal{P}$ samples $a \leftarrow \$\mathbb{Z}_q$ and computes $\alpha = g^a$, and sends α to $\mathcal{V}$;
- $\mathcal{V}$ samples $\beta \leftarrow \$\mathbb{Z}_q = \mathcal{B}_{pk,\lambda}$, and sends β to $\mathcal{P}$;
- $\mathcal{P}$ computes $\gamma = \beta x + a \mod q$, and sends γ to $\mathcal{V}$;
- $\mathcal{V}$ checks that $g^\gamma \cdot y^{-\beta} = \alpha$.

Completeness:

$$g^\gamma \cdot y^{-\beta} = g^{\beta x + a} \cdot g^{-\beta x} = g^{\cancel{\beta x} - \cancel{\beta x} + a} = g^a.$$

Schnorr protocol has a simulator: pick β, γ at random, let $\alpha = g^\gamma \cdot y^{-\beta}$, and output (α, β, γ). In a real execution, $\gamma = \beta x + a$ is uniform regardless of β, and α is the unique value such that $\alpha = g^\gamma \cdot y^{-\beta}$. ◇

SP holds under Discrete Log (DL) assumption. Assume $\mathcal{A}$ breaks SP with non negligible probability. $\mathcal{A}'$ gets $y = g^x$ for random $x \leftarrow \$\mathbb{Z}_q$, and sets $pk = y$ in a game with $\mathcal{A}$. $\mathcal{A}$ outputs $(\alpha, \beta, \gamma, \beta', \gamma')$, with $\beta \neq \beta'$ and $g^\gamma \cdot y^{-\beta} = \alpha = g^{\gamma'} \cdot y^{-\beta'}$. We get that

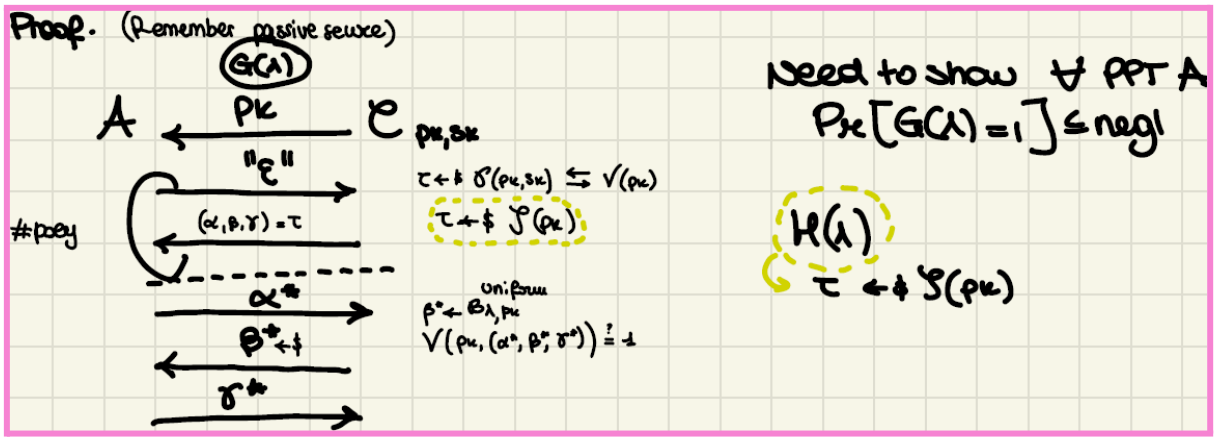
$$g^\gamma y^{-\beta} = g^{\gamma'} y_{-\beta'} \implies g^{\gamma - \gamma'} = y^{\beta - \beta'} \implies y = g^{(\gamma-\gamma')(\beta-\beta')^{-1}}$$

so $x = (\gamma - \gamma')(\beta - \beta')^{-1}$.

Proof that HVZK plus SS imply passive security

[Thm I - HVZK + SS → Passive Secure ID]

Starting from the definition of passive security, we define the experiments $G(\lambda)$ and $H(\lambda)$.



We need to show that $\forall PPT\ A,\ Pr[G(\lambda) = 1] \leq negl(\lambda)$. We use for the proof two lemmas.

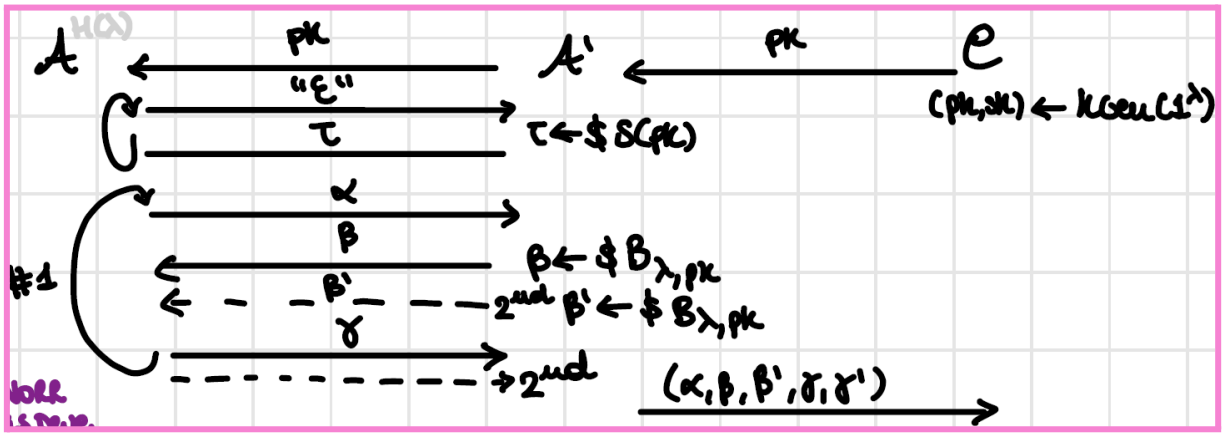
- [Lemma 1] $G(\lambda) \approx_c H(\lambda)$

Exercise. Prove it by HVZK.

- [Lemma 2] $Pr[H(\lambda) = 1] \leq negl(\lambda)$

This lemma is a reduction to SS. (Note: questa reduction è diversa dalle normali).

Assume not exist PPT A winning $H(\lambda)$ w.p. $\geq 1/poly(\lambda)$ → construct PPT A' against SS.



Let $\epsilon(\lambda) = Pr[H(\lambda) = 1] \geq 1/poly(\lambda)$.

Call z the state of A until it receives β, β' and $p_z = Pr[Z = z]$

→ then $Pr[H(\lambda) = 1] = \epsilon(\lambda) = \Sigma_z p_z \delta_z = IE[\delta_z]$, where $\delta_z = Pr[H(\lambda) = 1 | Z = z]$

Assume $|\mathcal{B}_{\lambda,pk}|^{-1} = negl(\lambda)$, so the probability that A' wins is →

$$Pr[A' wins] \geq Pr[A' wins \wedge (\beta \neq \beta')]$$

$$\geq Pr[A' wins \mid (\beta \neq \beta')] - 1/|\mathcal{B}_{\lambda,pk}| = \Sigma_z p_z \delta_z^2 - |\mathcal{B}_{\lambda,pk}|^{-1} = IE[\delta_z^2] - |\mathcal{B}_{\lambda,pk}|^{-1}$$

$$\geq IE[\delta_z^2] - |\mathcal{B}_{\lambda,pk}|^{-1} \text{ (Jensen's inequality)} = \varepsilon^2(\lambda) - |\mathcal{B}_{\lambda,pk}|^{-1}$$

(Schnorr, $1/poly^2 - 1/q = 1/poly$) Done!

- Thm I - HVZK + SS → Passive Secure ID (+ proof) + Lemma 1 (+ proof) + Lemma 2 (+ proof)

Syllabus 2022

Information-Theoretic Cryptography:

- Perfect secrecy, one-time pad, Shannon's theorem.
- Perfect authentication, universal hashing, extractors, leftover-hash lemma.

Computational Security:

- One-Way Functions (OWF) and complexity theory.
- Brush-up on number theory, candidate OWF (Factoring, RSA, DL).
- Computational indistinguishability, decisional assumptions (DDH).

Symmetric Cryptography:

- Pseudorandom Generators (PRG), hard-core bits, PRG constructions.
- Pseudorandom Functions (PRF), PRF constructions, Feistel networks.
- Symmetric encryption: Definitions and constructions, modes of operation.
- Message authentication: Definitions and constructions, authenticated encryption.
- Hash functions: Random oracle model, first/second pre-image resistance, collision resistance, Merkle-Damgaard construction.

Public-Key Cryptography:

- Public-key encryption: Definitions, RSA and ElGamal cryptosystems. Cramer-Shoup encryption.
- Digital signatures: Definitions, full-domain hash, signatures from OWF, Waters' signatures.
- Identification schemes: Definitions, constructions and applications to signatures.
- Identity-based encryption and applications.

Syllabus 2021

(...)

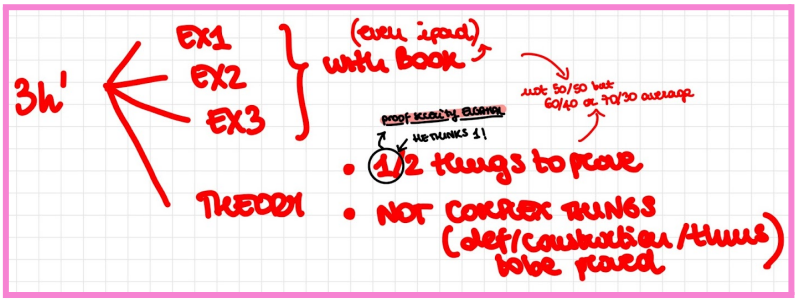
- Brush-up on number theory, candidate OWF (Factoring, RSA, QR, DL).
- Computational indistinguishability, decisional assumptions (DDH, QR).

(...)

Symbols

References Definitions Theorems Exercises
(•) → CRYPTO 2022
italiano/inglese

Info EXAM 2022



Definitions

- Def 1 - π is correct if $\forall k, \forall m \rightarrow Dec(k, Enc(k, m)) = m$
- Def 2 - Perfect Secrecy
- Def 3 - Information Theoretic MAC
- Def 4 - Pairwise Independent Hashing
- Def 5 - Min Entropy
- Def 6 - Seeded Extractor
- Def 7 - Polynomial
- Def 8 - Negligible
- Def 9 - PPT
- Def 10 - OWF
- Def 11 - PRG
- Def 12 (3A, 3B, 3C) - Security of PRG (No 2022)
- Def A - Computationally indistinguishable (*)
- Def 13 - OTS
- Def 14 - Hard-Core Predicate
- Def 15 - CPA-security
- Def 16 - PRF
- Def 17 - UF-CMA
- Def 18 - Almost Universality (2022, [7]: Definition 16 (Universal Hash Function))
- Def 19 - PRP (*)
- Def 20 - CCA-security
- Def 21 - Authenticity
- Def 22 - Collision Resistant Hash Function
- Def B - Trapdoor permutation (TDP) (*)
- Def C - CPA-security for PKE (*)
- Def 23 - RSA Assumption (*)
- Def 24 - CCA-security for PKE
- Def D - Secure signature
- Def E - HVZK (Honest Verifier Zero Knowledge)
- Def F - Sound

---

30 definitions

Theorems

- Thm 1 - Equivalent notions of perfect secrecy (+ proof)
- Thm 2 - One Time Pad is perfectly secret (+ proof)
- Thm 3 - Shannon (+ proof)
- Thm 4 - Any pairwise independent function is $\frac{1}{|\Phi|}$ - Statistically Secure (+ proof)
- (Construction +) Lemma - The functions in the family $\mathcal{H}$ (Construction) are pairwise independent (+ proof)
- Thm 5 - Any t-time $2^{-\lambda}$ - statistically secure MAC has key of size $(t+1)\lambda$
- Thm 6 - Leftover Hash Lemma (+ proof) + Lemma - Collision Probability (+ proof)
- (Construction +) Thm 7 - PRG⇒Secure encryption (+ proof) + Lemma 1 (+ proof) + Lemma 2 (+ proof)
- (Construction +) Thm 8 - If there exists PRG $G: \{0,1\}^{\lambda} \to \{0,1\}^{\lambda+1}$ then there exists PRG $G^{l}: \{0,1\}^{\lambda} \to \{0,1\}^{\lambda+l}$, $\forall l(\lambda) = poly(\lambda)$ (+ proof) + Lemma (+ proof)
- Thm A - Every OWF $f: \{0,1\}^{n} \to \{0,1\}^{n}$ has some HARD-CORE bits (*)
- Thm 9 - Goldreich-Levin + Corollario - Let f be a OWP and g, h be as in the theorem, then $G(s) = (g(s), h(s))$ is a PRG with $l(\lambda) = 1$ (+ proof)

  or Thm B - Let $f: \{0,1\}^{n} \to \{0,1\}^{n}$ be a OWP. Then $G(s) = f(s)||h(s)$ is a PRG with $l = 1$ (+ proof) (*)
- Thm 10 - OWFs⇒CPA-secure SKE (*)
- (Construction +) Thm 11 - If $\mathcal{F}$ is a PRF family, then Π above is CPA-secure (+ proof) + Lemma 1 (+ proof) + Lemma 2 (+ proof)
- (Construction +) Thm 12 - If G is a PRG, F is a PRF
- Thm 13 - PRF⇒MAC, let $\mathcal{F} = \{F_k\{0.1\}^{n} \to \{0.1\}^{l}\}_{k\epsilon\{0.1\}^{\lambda}}$ be a PRF family, then $Tag(k,m) = F_k(m)$ is UF-CMA, so long as $l(\lambda) = \boldsymbol{\omega}(log(\lambda))$ (+ proof) + Lemma 1A (+ proof as Exercise) or Lemma 1B (+ proof as Exercise) + Lemma 2B (+ proof)

  or Thm C - PRF⇒MAC, let $\mathcal{F} = \{F_k\{0.1\}^{n} \to \{0.1\}^{l}\}_{k\epsilon\{0.1\}^{\lambda}}$ be a PRF family, then $Tag(k,m) = F_k(m)$ is UF-CMA (+ proof) (*)
- Thm 14 - $\mathcal{F}(\mathcal{H}) = \{F_k(h_s(\cdot))\}$ is a PRF from $N \to l$, assuming $\mathcal{H}$ is AU and $\mathcal{F}$ is a PRF from $n \to l$ (+ proof) + Lemma 1 (+ proof as Exercise) + Lemma 2 (+ proof) (2022, [7]: Theorem 14)
- Lemma - CBC-MAC defines completely an AU family
- Thm 15 - CBC-MAC directly gives a FIXED INPUT LENGTH MAC (2022, [7]: Theorem 15 (CBC-MAC is a computationally secure AU hash function if F is a PRF), Theorem 16 (CBC-MAC is a PRF), Theorem 17 (CBC-MAC is AU))
- Thm 16 - If $\mathcal{H}$ is ε-AXU and $\mathcal{F}$ is a PRF, then XOR-MAC is UF-CMA (*)
- Thm 17 - (Luby-Rackoff) If $\mathcal{F}$ is a PRF, $\psi_{\mathcal{F}}[3]$ is a PRP (+ proof) + Lemma 1 (+ proof) + Lemma 2 (+ proof as Exercise) + Lemma 3 (+ proof) + Lemma 4 (+ proof) (*)
- Thm 18 - $\psi_{\mathcal{F}}[4]$ is a strong PRP (No 2022)
- Thm 19 - Assuming $\mathcal{F}$ is a PRF family, then CTR mode yields a CPA-secure VIL SKE (+ proof) + Lemma 1 (+ proof) + Lemma 2 (+ proof) (*)

- Thm 20 - CPA + Auth ⇒ CCA (+ proof as Exercise) (*)
- Thm 21 - Encrypt-then-MAC is CCA secure, assuming Π_1 is CPA-secure and Π_2 is STRONG UF-CMA (+ proof) (*)
- Thm 22 - The MD construction gives a CRH $\mathcal{H}'$ from $l'(\lambda) \to n(\lambda)$ assuming $\mathcal{H}$ is CRH from $n + 1 \to n$ (+ proof)
- Thm 23 - The strengthen MD is CR for VIL inputs (+ proof)
- Lemma - If $gcd(a, n) \neq 1$, then a not invertible (+ proof)
- Lemma - Let $a, b \ s.t. \ a \geq b > 0$ then $gcd(a, b) = gcd(b, a \bmod b)$ (+ proof)
- Thm 24 - Given a, b we can compute $gcd(a, b)$ in polynomial time ($O(log^3 \lambda)$) and also we can find $u, v \ s.t. \ gcd(a, b) = ua + bv$ (+ proof + CLAIM with proof)
- Thm 25 - Prime Number Theorem
- Thm 26 - We can test if a λ-bit integer is prime in $poly(\lambda)$ time
- Thm 27 - Lagrange's theorem + Corollario with Euler's theorem (+ proof) (*)
- Thm D - DDH $\to G_{g,q}$ is a PRG (*)
- Thm 28 - DDH $\to G^t_{g,q}$ is a PRG for $t = poly(\lambda)$ (+ proof as Exercise)
- Thm 29 - $\mathcal{F}_{NR}$ is a PRF family under DDH (+ proof)
- Thm E - Assuming DL, above construction is CRH (+ proof) (*)
- Thm 30 - TDP → CPA PKE (+ proof) (*)
- Thm 31 - Under DDH, the Elgamal PKE is CPA secure (+ proof) + Lemma 1 (+ proof) + Lemma 2 (+ proof as Exercise)
- Thm F - CSlite is CCA1 secure under DDH + Lemma 1 (+ proof) + Lemma 2 (+ proof + CLAIM 1 with proof + CLAIM 2 with proof)
- Thm G - The above scheme (FDH) is UF-CMA, assuming TDP is one way and H modeled as a random oracle (+ proof)
- Thm H - FS yields UF-CMA signature in the ROM, assuming ID scheme is passively secure (and non-degenerate) (+ proof)
- Thm I - HVZK + SS → Passive Secure ID (+ proof) + Lemma 1 (+ proof) + Lemma 2 (+ proof)

---

42 theorems

(*) → Look above CRYPTO 2022

Introduction and secure communication

Symmetric Cryptography

- Def 1 - π is correct if $\forall k, \forall m \rightarrow Dec(k, Enc(k, m)) = m$

Perfect secrecy

- Def 2 - Perfect Secrecy
- Thm 1 - Equivalent notions of perfect secrecy (+ proof)

One time pad (OTP)

- Thm 2 - One Time Pad is perfectly secret (+ proof)

Shannon's theorem

- Thm 3 - Shannon (+ proof)

Message Authentication Codes (MACs)

Statistically-secure (one-time) MACs

- Def 3 - Information Theoretic MAC

Pairwise Independent Hashing

- Def 4 - Pairwise Independent Hashing
- Thm 4 - Any pairwise independent function is $\frac{1}{|\Phi|}$ - Statistically Secure (+ proof)
- (Construction +) Lemma - The functions in the family $\mathcal{H}$ (Construction) are pairwise independent (+ proof)

Limits of statistically-secure MACs.

- Thm 5 - Any t-time $2^{-\lambda}$ - statistically secure MAC has key of size $(t + 1)\lambda$

Randomness Extraction

Von Neumann extractor

Definition of min-entropy and seedless extractors

- Def 5 - Min Entropy

Definition of seeded extractors

- Def 6 - Seeded Extractor

Leftover-hash lemma: Statement and proof

- Thm 6 - Leftover Hash Lemma (+ proof) + Lemma - Collision Probability (+ proof)

Computational security

- Def 7 - Polynomial
- Def 8 - Negligible
- Def 9 - PPT

One-Way Functions (OWF)

- Def 10 - OWF

Impagliazzo's worlds

Pseudorandomness

PRG

- Def 11 - PRG
- Def 12 (3A, 3B, 3C) - Security of PRG (No 2022)
- Def A - Computationally indistinguishable (*)

Definition of one-time computational security for SKE and construction from any PRG

- Def 13 - OTS
- (Construction +) Thm 7 - PRG⇒Secure encryption (+ proof) + Lemma 1 (+ proof) + Lemma 2 (+ proof)

Proof that one-bit stretch is sufficient to obtain arbitrary polynomial stretch

- (Construction +) Thm 8 - If there exists PRG $G: \{0,1\}^{\lambda} \rightarrow \{0,1\}^{\lambda+1}$ then there exists PRG $G^{l}: \{0,1\}^{\lambda} \rightarrow \{0,1\}^{\lambda+l}, \forall l(\lambda) = poly(\lambda)$ (+ proof) + Lemma (+ proof)

Proof that OWPs imply PRGs with one-bit stretch

- Def 14 - Hard-Core Predicate
- Thm A - Every OWF $f:\{0,1\}^n \to \{0,1\}^n$ has some HARD-CORE bits (*)
- Thm 9 - Goldreich-Levin + Corollario - Let f be a OWP and g, h be as in the theorem, then $G(s) = (g(s), h(s))$ is a PRG with $l(\lambda) = 1$ (+ proof)

  or Thm B - Let $f:\{0,1\}^n \to \{0,1\}^n$ be a OWP. Then $G(s) = f(s)||h(s)$ is a PRG with $l = 1$ (+ proof) (*)

CPA secure SKE

CPA security

- Def 15 - CPA-security
- Thm 10 - OWFs⇒CPA-secure SKE (*)

PRF

- Def 16 - PRF

Construction of CPA secure SKE from any PRF family

- (Construction +) Thm 11 - If $\mathcal{F}$ is a PRF family, then Π above is CPA-secure (+ proof) + Lemma 1 (+ proof) + Lemma 2 (+ proof)

Constructing PRFs from PRGs: The GGM construction

- (Construction +) Thm 12 - If G is a PRG, F is a PRF

Message Authentication Codes (2)

Computationally secure MACs

- Def 17 - UF-CMA

Every PRF yields a FIL MAC

- Thm 13 - PRF⇒MAC, let $\mathcal{F} = \{F_k\{0.1\}^n \to \{0.1\}^l\}_{k\epsilon\{0.1\}^\lambda}$ be a PRF family, then $Tag(k,m) = F_k(m)$ is UF-CMA, so long as $l(\lambda) = \boldsymbol{\omega}(log(\lambda))$ (+ proof) + Lemma 1A (+ proof as Exercise) or Lemma 1B (+ proof as Exercise) + Lemma 2B (+ proof)

  or Thm C - PRF⇒MAC, let $\mathcal{F} = \{F_k\{0.1\}^n \to \{0.1\}^l\}_{k\epsilon\{0.1\}^\lambda}$ be a PRF family, then $Tag(k,m) = F_k(m)$ is UF-CMA (+ proof) (*)

Domain extension for PRFs via almost-universal hash functions

- Def 18 - Almost Universality (2022, [7]: Definition 16 (Universal Hash Function))
- Thm 14 - $\mathcal{F}(\mathcal{H}) = \{F_k(h_s(\cdot))\}$ is a PRF from $N \to l$, assuming $\mathcal{H}$ is AU and $\mathcal{F}$ is a PRF from $n \to l$ (+ proof) + Lemma 1 (+ proof as Exercise) + Lemma 2 (+ proof)
  (2022, [7]: Theorem 14)

Constructions of almost universal hash functions

CBC-MAC and Encrypted CBC-MAC

- Lemma - CBC-MAC defines completely an AU family
- Thm 15 - CBC-MAC directly gives a FIXED INPUT LENGTH MAC
  (2022, [7]: Theorem 15 (CBC-MAC is a computationally secure AU hash function if F is a PRF), Theorem 16 (CBC-MAC is a PRF), Theorem 17 (CBC-MAC is AU))

XOR-MAC

- Thm 16 - If $\mathcal{H}$ is ε-AXU and $\mathcal{F}$ is a PRF, then XOR-MAC is UF-CMA (*)

MAC: FIL vs VIL

PRP

- Def 19 - PRP (*)

Feistel networks

- Thm 17 - (Luby-Rackoff) If $\mathcal{F}$ is a PRF, $\psi_{\mathcal{F}}[3]$ is a PRP (+ proof) + Lemma 1 (+ proof) + Lemma 2 (+ proof as Exercise) + Lemma 3 (+ proof) + Lemma 4 (+ proof) (*)

- Thm 18 - $\psi_{\mathcal{F}}[4]$ is a strong PRP (No 2022)

Domain extension for SKE

Modes of operation: ECB, CFB, CBC and CTR

- Thm 19 - Assuming $\mathcal{F}$ is a PRF family, then CTR mode yields a CPA-secure VIL SKE (+ proof) + Lemma 1 (+ proof) + Lemma 2 (+ proof) (*)

Authenticated encryption and its relation to CCA security

CCA security

- Def 20 - CCA-security

Authenticity

- Def 21 - Authenticity
- Thm 20 - CPA + Auth ⇒ CCA (+ proof as Exercise) (*)

Combining SKE and MAC schemes

Encrypt-then-MAC and proof of CCA security

- Thm 21 - Encrypt-then-MAC is CCA secure, assuming Π_1 is CPA-secure and Π_2 is STRONG UF-CMA (+ proof) (*)

(RECAP) From minicrypt to cryptomania

Collision Resistant Hashing

- Def 22 - Collision Resistant Hash Function

The Merkle-Damgaard and Merkle tree constructions

- Thm 22 - The MD construction gives a CRH $\mathcal{H}'$ from $l'(\lambda) \to n(\lambda)$ assuming $\mathcal{H}$ is CRH from $n + 1 \to n$ (+ proof)
- Thm 23 - The strengthen MD is CR for VIL inputs (+ proof)

Constructing secure compression functions

Number theory

Modular arithmetic

- Lemma - If $gcd(a, n) \neq 1$, then a not invertible (+ proof)

Euclidean algorithm

- Lemma - Let $a, b \ s.t. \ a \geq b > 0$ then $gcd(a, b) = gcd(b, a \bmod b)$ (+ proof)
- Thm 24 - Given a, b we can compute $gcd(a, b)$ in polynomial time ($O(log^3 \lambda)$) and also we can find $u, v \ s.t. \ gcd(a, b) = ua + bv$ (+ proof + CLAIM with proof)

Modular exponentiation

Prime numbers

- Thm 25 - Prime Number Theorem
- Thm 26 - We can test if a λ-bit integer is prime in $poly(\lambda)$ time

Factorization

Lagrange's theorem

- Thm 27 - Lagrange's theorem + Corollario with Euler's theorem (+ proof) (*)

Cyclic group

The Discrete Logarithm (DL) problem

Diffie-Hellman key exchange

CDH and DDH

Simple number-theoretic constructions

OWFs, PRGs and PRFs from DDH

- Thm D - DDH → $G_{g,q}$ is a PRG (*)
- Thm 28 - DDH → $G_{g,q}^t$ is a PRG for $t = poly(\lambda)$ (+ proof as Exercise)
- Thm 29 - $\mathcal{F}_{NR}$ is a PRF family under DDH (+ proof)

CRH from DL

- Thm E - Assuming DL, above construction is CRH (+ proof) (*)

Public key encryption

TDP (Trapdoor Permutation)

- Def B - Trapdoor permutation (TDP) (*)

CPA security for PKE

- Def C - CPA-security for PKE (*)

CPA PKE from TDP

- Thm 30 - TDP⇒CPA PKE (+ proof)

RSA (Rivest–Shamir–Adleman)

RSA assumption

- Def 23 - RSA Assumption (*)

PKCS #1 v1.5 and PKCS #2 v2.0

Rabin's TDP and its equivalence to Factoring

Elgamal PKE and its CPA security from the DDH assumption

- Thm 31 - Under DDH, the Elgamal PKE is CPA secure (+ proof) + Lemma 1 (+ proof) + Lemma 2 (+ proof as Exercise)

CCA security for PKE

- Def 24 - CCA-security for PKE

Cramer-Shoup encryption

- Thm F - CSlite is CCA1 secure under DDH + Lemma 1 (+ proof) + Lemma 2 (+ proof + CLAIM 1 with proof + CLAIM 2 with proof)

Digital signature

- Def D - Secure signature

Public key infrastructure and Identity Based Encryption

Signing with RSA

FDH (Full Domain Hash)

- Thm G - The above scheme (FDH) is UF-CMA, assuming TDP is one way and H modeled as a random oracle (+ proof)

Identification (ID) schemes

Canonical ID schemes (Σ-protocol)

Fiat-Shamir signatures

- Thm H - FS yields UF-CMA signature in the ROM, assuming ID scheme is passively secure (and non-degenerate) (+ proof)

Honest Verifier Zero Knowledge (HVZK) and Special Soundness (SS) for canonical ID schemes

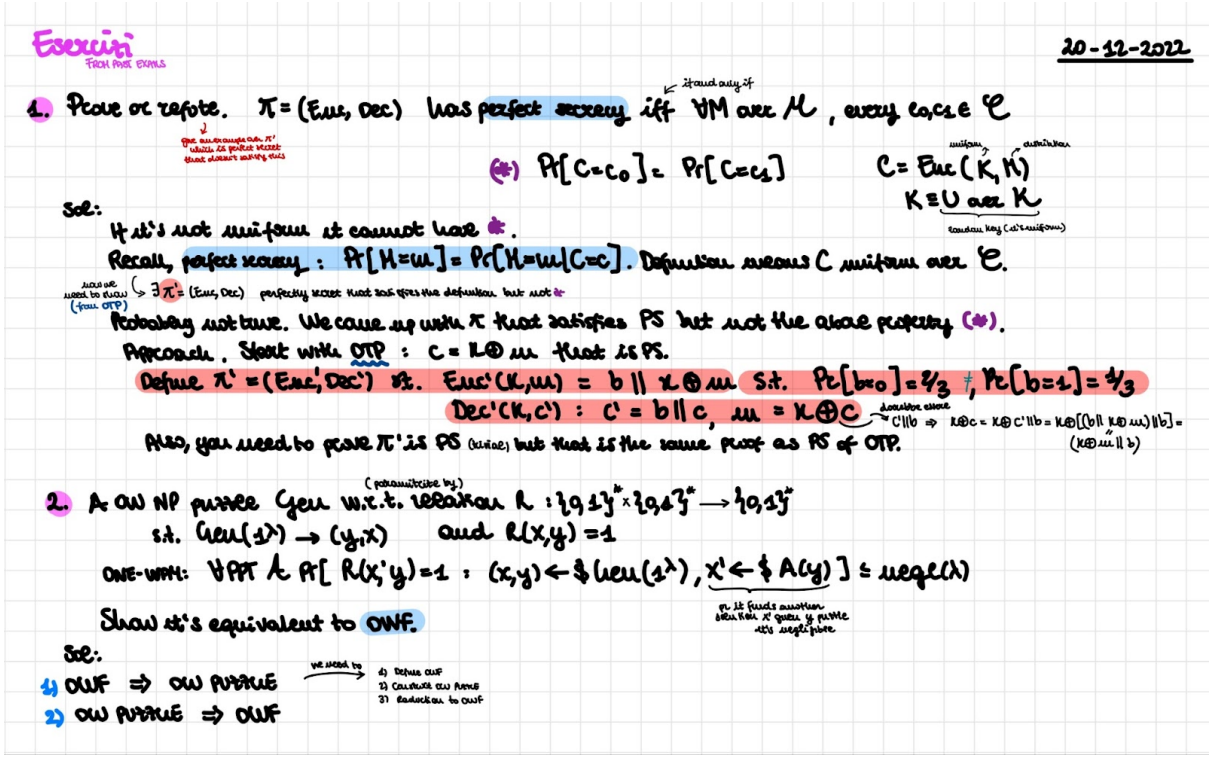
- Def E - HVZK (Honest Verifier Zero Knowledge)
- Def F - Sound

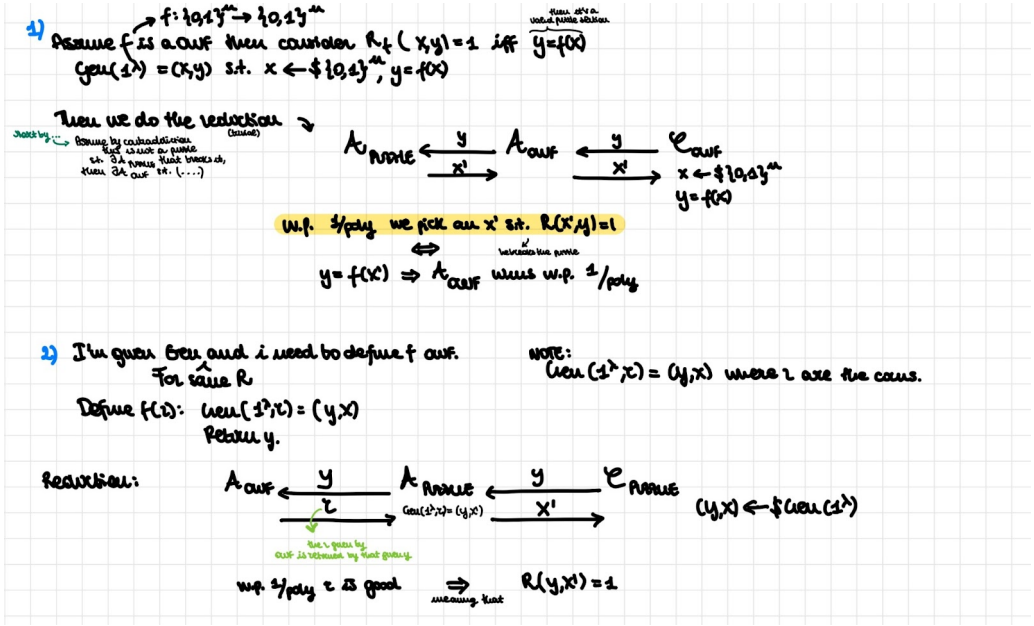
Proof that HVZK plus SS imply passive security

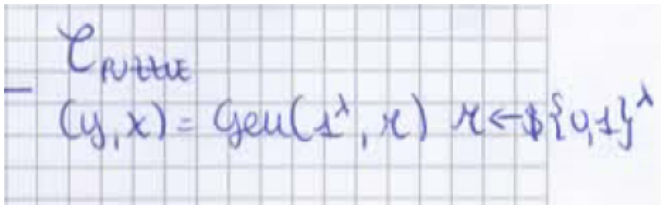
- Thm I - HVZK + SS → Passive Secure ID (+ proof) + Lemma 1 (+ proof) + Lemma 2 (+ proof)

Exercises

Lecture 23 (2022)

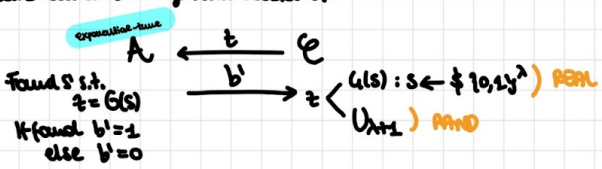






3. $G: \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ a PRG. Show that no such G can be secure against some unbounded adversary.

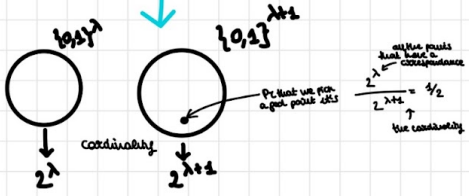
sol: Create an adversary that breaks G (breaks G)



$Pr[REAL_{G,A}(\lambda)=1] = Pr[b'=1 \text{ s.t. } z=G(s), s \leftarrow \$ \{0,1\}^\lambda] = 1$

$Pr[RAND_{G,A}(\lambda)=1] = Pr[b'=1 \text{ s.t. } z \leftarrow \$ U_{\lambda+1}] \leq \frac{2^\lambda}{2^{\lambda+1}} = 1/2$ ← negl

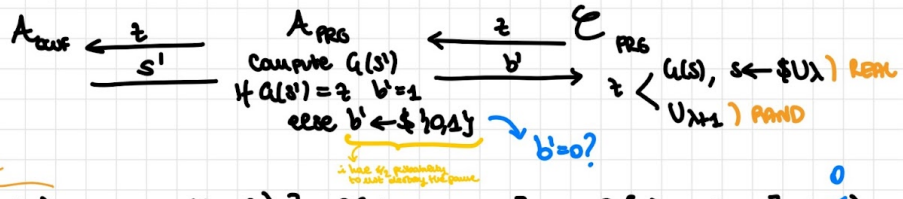
$|Pr[REAL_{G,A}(\lambda)=1] - Pr[RAND_{G,A}(\lambda)=1]| \geq 1 - 1/2 = 1/2$ ← inverse poly



4. $G: \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ a PRG. Show G is a owf.

By reduction:

Assume, w.l.o.g., an adversary that ...



$Pr[A(G(s)) = 1 : s \leftarrow \$ \{0,1\}^\lambda] = Pr[A_{owf} \text{ wins}] \cdot 1 + Pr[A_{owf} \text{ loses}] \cdot 1/2 \geq$

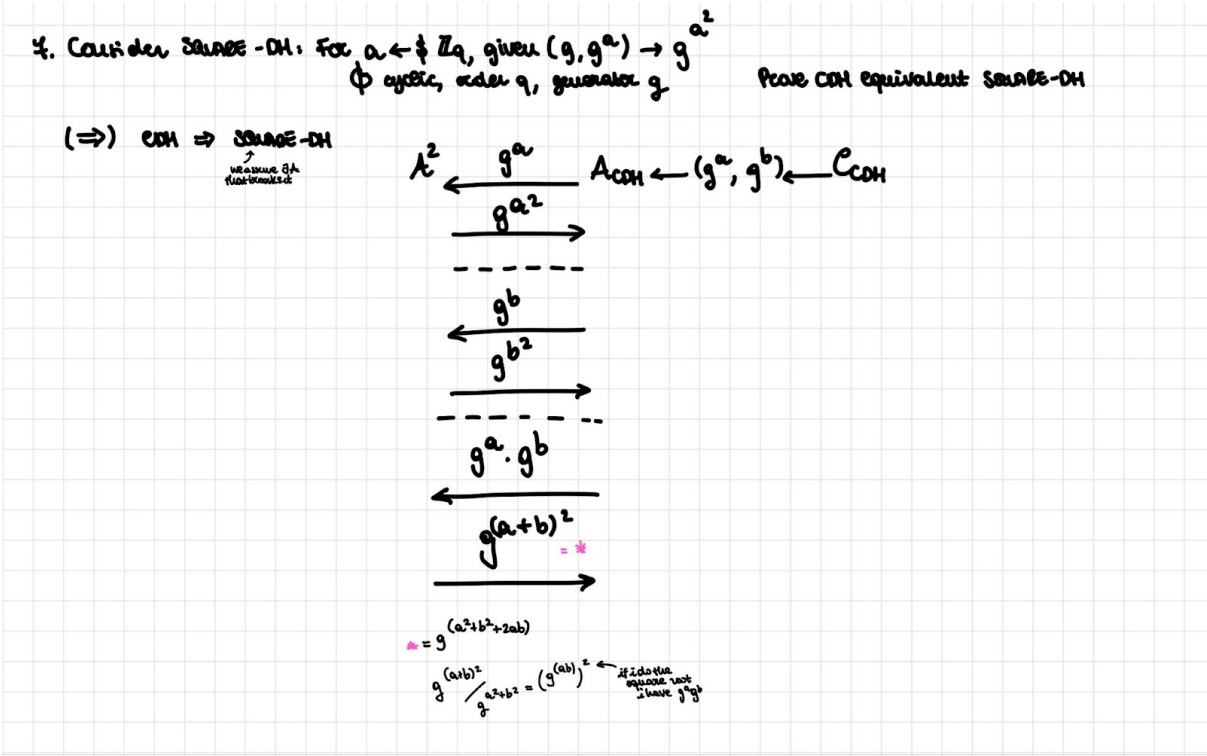
$\varepsilon(\lambda) + 1/2 (1 - \varepsilon(\lambda)) = 1/2 + \frac{\varepsilon(\lambda)}{2} = \varepsilon(\lambda)$

$\varepsilon(\lambda)$ ← it becomes $\varepsilon - 1/2$ → not 1/poly

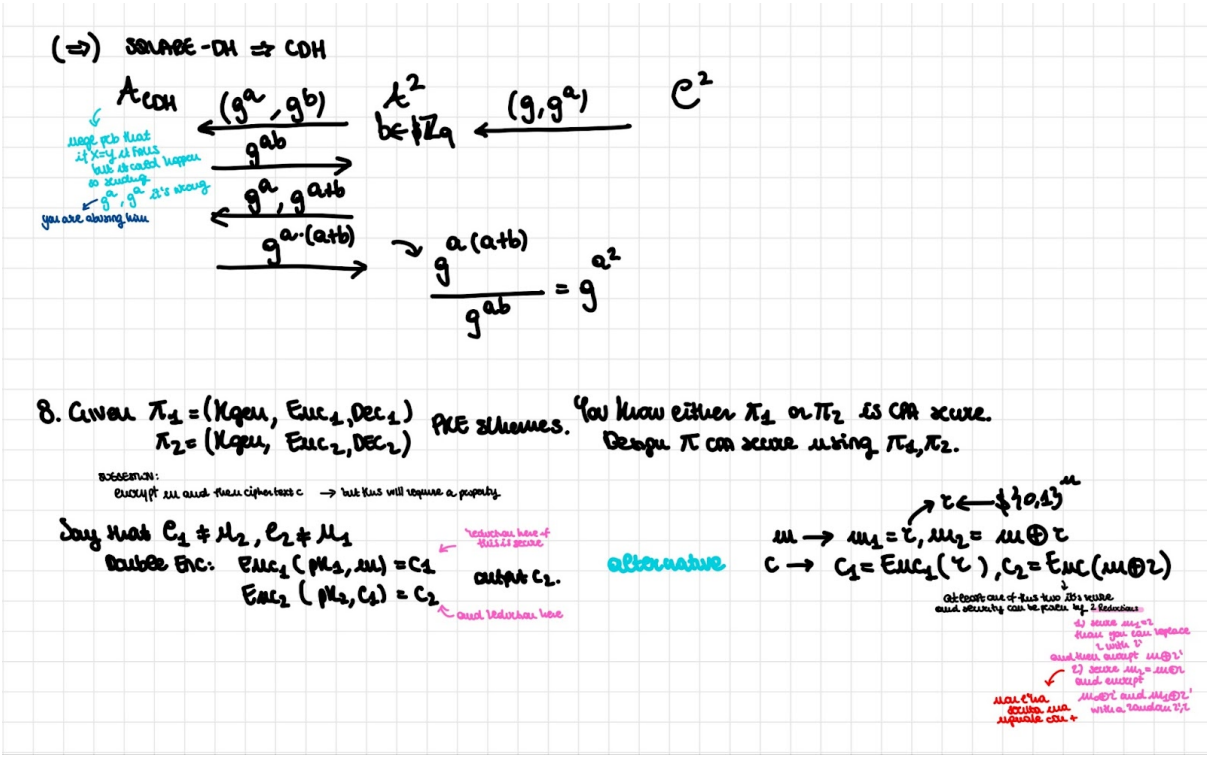
$Pr[A_{PRG}(z)=1 : z \leftarrow \$ U_{\lambda+1}] \leq 1/2$ ← at most 1/2 the probability it finds a pre-image

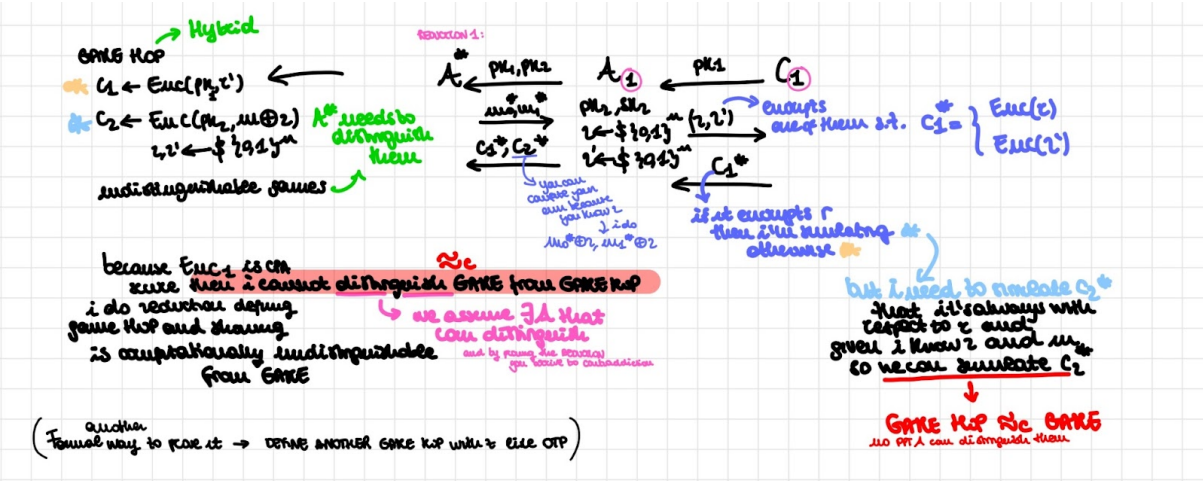
$|Pr[A(G(s)) = 1 : s \leftarrow \$ \{0,1\}^\lambda] - Pr[A_{PRG}(z)=1 : z \leftarrow \$ U_{\lambda+1}]| \geq \frac{\varepsilon(\lambda)}{2}$

$1/2 + \frac{\varepsilon(\lambda)}{2}$ − $1/2$ ← or pri 1/2



Vedi Lecture 14, pag 71, in cui parla di DDH, CDH e DL → proof simile!





Exams

Recap

2018

- Febbraio
    1. PRGs with Weak Seeds (No $H_2(S)$ nel 2022)
    2. Selective Unforgeability (=)
    3. Actively Secure ID Schemes
- Giugno
    1. Seeded Extraction
    2. CBC Mode and CCA Security
    3. Strong Unforgeability
- Luglio
    1. PRGs as OWFs
    2. OFB Mode and CCA Security
    3. Non-Adaptive and Weak Unforgeability

2019

- Gennaio
    1. MACs Combiners
    2. Replayable security (=)
    3. +3-DDH (=)
- Febbraio
    1. Hash Functions Combiners (*)
    2. Multi-Message CPA Security (=)
    3. Weak PRFs (=)

2020

- Gennaio
    1. Hash combiners (=)
    2. Weak PRFs
    3. Selective Unforgeability (*)
- Febbraio
    1. PRFs Combiners

    2. Replayable CCA security (=)
    3. +1-DDH (=)
- Maggio
    1. A PRG candidate
    2. PKE Combiners (*, No IND-CPA nel 2022)
    3. ID scheme based on RSA (=)

2021

- Gennaio
    1. Weak collision resistant
    2. PK-only security (=)
    3. A OWF based on DLOG
- Luglio
    1. Two-time Perfect secrecy
    2. OWF or not?
    3. IBE Combiner (No IBE nel 2022)
- Settembre
    1. MAC design
    2. Collision Resistant or Not
    3. Discrete Log Attack (=)

2022

- Gennaio
    1. A simple PRG
    2. PKE combiners (*)
    3. Amplifying a discrete LOG attack (=)
- Febbraio
    1. PRP Combiners
    2. Selective Unforgeability (=)
    3. ID Scheme based on RSA (=)
- Giugno
    1. A simple PRF
    2. PK-Only Security (=)
    3. CDH Variants
- Luglio
    1. Collision Resistant or Not?
    2. Multi-Message CPA Security (=)
    3. A Variant of CDH
- Settembre
    1. Chain encryption
    2. MACs with Multi-key Security
    3. Alternative DDH Characterization
- Ottobre
    1. A PRF construction
    2. Circular security
    3. A signature construction

2023

- Gennaio
    1. PRG Candidates
    2. CCA Secure Or Not?

3. RSA Variants
4. (*Theory*) Collision-Resistant hashing

- Febbraio
  1. PRGs and Complexity Theory
  2. CPA-to-CCA Transform
  3. Derandomizing Signatures
  4. (*Theory*) Pseudorandom Permutations

Note: (*) → similar exercise, (=) equal to other exercise (or included)

Testi

2018

- Febbraio
  1. PRGs with Weak Seeds (No $H_2(S)$ nel 2022)

1 PRGs with Weak Seeds **10 Points**

Let $G: \{0,1\}^m \to \{0,1\}^{2m}$ be a $(t_{\mathsf{prg}}, \varepsilon_{\mathsf{prg}})$-secure PRG. Explain how to safely use G in a setting where the seed S, instead of being uniform, is such that $\mathbb{H}_2(S) \geq m - d$. Assuming $m = 128$ and $d = 8$, show how to choose the parameters $t_{\mathsf{prg}}, \varepsilon_{\mathsf{prg}}$ in such a way that your construction achieves security 2^{-80} against all adversaries running in time at most 2^{20}.

  2. Selective Unforgeability

2 Selective Unforgeability **10 Points**

Define a variant of universal unforgeability against chosen-message attacks (UF-CMA) for digital signatures, in which the adversary has to commit to the message $m^* \in \mathcal{M}$ on which he will forge a signature before seeing the public key (where $\mathcal{M}$ is the message space); note that in the definition the adversary should still be allowed to sign arbitrary messages. Name your notion SF-CMA (i.e., selective unforgeability against chosen-message attacks). Prove or disprove:

(a) UF-CMA $\Rightarrow$ SF-CMA.

(b) SF-CMA $\Rightarrow$ UF-CMA.

  3. Actively Secure ID Schemes

3 Actively Secure ID Schemes **10 Points**

Let $\Pi = (\mathsf{Gen}, \mathcal{P}, \mathcal{V})$ be an ID scheme. Informally, an ID scheme is actively secure if no efficient adversary $\mathcal{A}$ (given just the public key pk) can make $\mathcal{V}$ accept, even after $\mathcal{A}$ participates maliciously in polynomially many interactions with $\mathcal{P}$ (given both the public

key pk and the secret key sk). More formally, we say that Π satisfies active security if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ there is a negligible function $\nu : \mathbb{N} \to [0,1]$ such that for any polynomial $n := n(\lambda)$ the following holds:

$$\Pr\left[\mathsf{out}_{\mathcal{V}}(\mathcal{A}_2(pk, s_n) \rightleftarrows \mathcal{V}(pk)) = 1 : \begin{array}{c}(pk, sk) \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda); s_0 := \varepsilon \\ (\forall i \in [n]) s_i \leftarrow (\mathcal{P}(pk, sk) \rightleftarrows \mathcal{A}_1(pk, s_{i-1}))\end{array}\right] \leq \nu(\lambda),$$

where s_0 is the empty string, $s_i \in \{0,1\}^*$ is some arbitrary state information, and where the probability is taken over the random coin tosses of algorithms Gen, $\mathcal{A}$, and $\mathcal{P}$. Answer the following questions.

(a) Let $\Pi' = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme, with message space $\mathcal{M}$. Prove that if Π' is CCA secure, the following ID scheme Π (based on Π') achives active security:

$\mathsf{Gen}(1^\lambda)$: Run $(pk, sk) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$ and output (pk, sk).

$\mathcal{P}(pk, sk) \rightleftarrows \mathcal{V}(pk)$: The verifier picks random $m \leftarrow\!\!\$\ \mathcal{M}$, and forwards $c \leftarrow\!\!\$\ \mathsf{Enc}(pk, m)$ to the prover. The prover replies with $m' = \mathsf{Dec}(sk, c)$, and finally the verifier accepts if and only if $m' = m$.

(b) Is the above protocol honest-verifier zero-knowledge? Prove your answer.

- Giugno
  1. Seeded Extraction

1 Seeded Extraction — 10 Points

Say that X is a (k, n)-source if $X \in \{0,1\}^n$, and the min-entropy of X is at least k. Explain how we can extract ℓ bits of uniform randomness from any (k, n)-source without relying on any computational assumption, and why a uniform seed is needed for this purpose. Then answer the following questions:

(a) Suppose that $\ell = 128$; what is the minimal amount of min-entropy needed in order to obtain statistical error $\varepsilon = 2^{-80}$? What is the entropy loss?

(b) Suppose that $k = 238$; what is the maximal amount of uniform randomness that you can obtain with statistical error $\varepsilon = 2^{-80}$? Explain how to obtain $\ell = 320$ using computational assumptions.

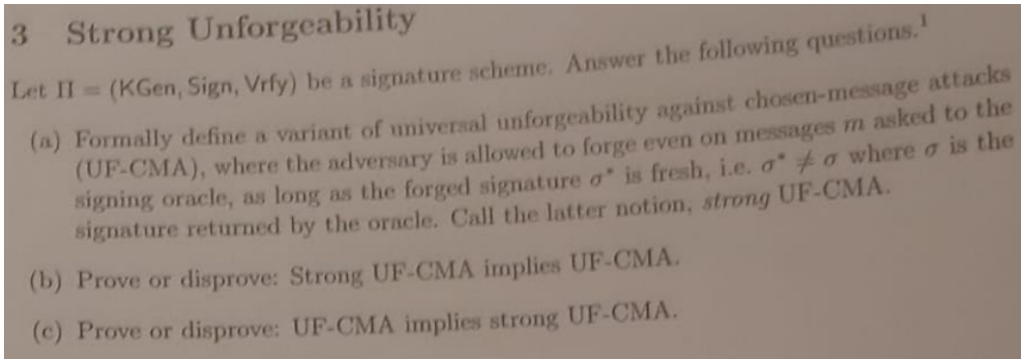
  2. CBC Mode and CCA Security

2 CBC Mode and CCA Security — 10 Points

Recall the CBC mode of operation: Given a message $m = (m_1, \ldots, m_t)$ consisting of t blocks $m_i \in \{0,1\}^n$, and a random key $\kappa \in \{0,1\}^\lambda$, the ciphertext is $c = (c_0, c_1, \ldots, c_n)$ where $c_0 \leftarrow\!\!\$\ \{0,1\}^n$, $c_i = P_\kappa(c_{i-1} \oplus m_i)$ for all $i \in [n]$, and $P : \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^n$ is a secure pseudorandom permutation.

In class, we proved that CBC mode yields a CPA-secure secret-key encryption scheme. Show that CBC mode is *not* CCA secure.

  3. Strong Unforgeability



3 Strong Unforgeability

Let $\Pi = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a signature scheme. Answer the following questions.[1]

(a) Formally define a variant of universal unforgeability against chosen-message attacks (UF-CMA), where the adversary is allowed to forge even on messages m asked to the signing oracle, as long as the forged signature σ^* is fresh, i.e. $\sigma^* \neq \sigma$ where σ is the signature returned by the oracle. Call the latter notion, *strong* UF-CMA.

(b) Prove or disprove: Strong UF-CMA implies UF-CMA.

(c) Prove or disprove: UF-CMA implies strong UF-CMA.

- Luglio
    1. PRGs as OWFs

1 PRGs as OWFs 10 Points

Let $G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ be a PRG with one-bit stretch. Prove that G is by itself a one-way function.

2. OFB Mode and CCA Security

2 OFB Mode and CCA Security 10 Points

Recall the OFB mode of operation: Given a message $m = (m_1, \ldots, m_t)$ consisting of t blocks $m_i \in \{0,1\}^n$, and a random key $\kappa \in \{0,1\}^\lambda$, the ciphertext is $c = (r_0, c_1, \ldots, c_t)$ where $r_0 \leftarrow\!\!\$\ \{0,1\}^n$, $r_i = F_\kappa(r_{i-1})$, and $c_i = r_i \oplus m_i$ for all $i \in [t]$, and $F : \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^n$ is a secure pseudorandom function.

In class, we mentioned that OFB mode yields a CPA-secure secret-key encryption scheme. Show that OFB mode is *not* CCA secure.

3. Non-Adaptive and Weak Unforgeability

3 Non-Adaptive and Weak Unforgeability 10 Points

Let $\Pi = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a signature scheme. Answer the following questions. (For each question asking to prove/disprove an implication between two notions, if you think the implication holds you must show a reduction from one definition to the other; on the other hand, if you think the implication does not hold, you must exhibit a scheme which satisfies one definition but not the other.)

(a) Formally define a variant of universal unforgeability against chosen-message attacks (UF-CMA), where the adversary is given (together with the public key) $q \in \mathsf{poly}(\lambda)$ message/signature pairs $(m_i, \sigma_i)_{i\in[q]}$, where the messages $(m_i)_{i\in[q]}$ are chosen by the adversary non-adaptively (i.e., all the same time) before obtaining the public key. As in UF-CMA, in order to win the game, the adversary then needs to forge on a message m^* which is fresh (i.e., not equal to any of the messages $m_1, \ldots, m_q$). Call the latter notion UF-naCMA.

(b) Formally define a variant of universal unforgeability against chosen-message attacks (UF-CMA), where the adversary is given (together with the public key) $q \in \mathsf{poly}(\lambda)$ message/signature pairs $(m_i, \sigma_i)_{i\in[q]}$, where each of the messages $(m_i)_{i\in[q]}$ is drawn uniformly at random from the message space. As in UF-CMA, in order to win the game, the adversary then needs to forge on a message m^* which is fresh (i.e., not equal to any of the messages $m_1, \ldots, m_q$). Call the latter notion UF-RMA.

(c) Prove or disprove: UF-naCMA implies UF-RMA.

(d) Prove or disprove: UF-naCMA implies UF-CMA.

2019

- Gennaio
    1. MACs Combiners

1 MACs Combiner 10 Points

Let $\Pi_1 = \mathsf{Tag}_1$, $\Pi_2 = \mathsf{Tag}_2$, and $\Pi_3 = \mathsf{Tag}_3$ be three deterministic MACs over key space $\mathcal{K} = \{0,1\}^\lambda$. You know that at least one of Π_1, Π_2, Π_3 is UF-CMA, but you don't know which one. Show how to design a secure MAC $\Pi^\star = \mathsf{Tag}^\star$ over key space $\mathcal{K}^3$ by combining Π_1, Π_2, and Π_3. Formally prove security of your candidate construction.

2. Replayable security

2 Replayable Security 10 Points

Let $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption (PKE) scheme. Recall the game $\mathbf{G}^{\mathsf{pke\text{-}cca}}_{\Pi,\mathcal{A}}(\lambda, b)$ defining CCA security, involving an adversary $\mathcal{A}$:

1. The challenger picks $(pk, sk) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$, and forwards pk to $\mathcal{A}$.
2. $\mathcal{A}$ can ask polynomially many decryption queries. Upon input a query c, the challenger returns $m = \mathsf{Dec}(sk, c)$ to $\mathcal{A}$.
3. $\mathcal{A}$ chooses two messages m_0, m_1 of equal length, and receives $c^\star \leftarrow\!\!\$\ \mathsf{Enc}(pk, m_b)$.
4. $\mathcal{A}$ can keep asking decryption queries, provided that these queries are different from the challenge ciphertext $c^\star$ defined in step 3.
5. $\mathcal{A}$ outputs a bit b'.

We say that a PKE scheme Π is CCA-secure if $\{\mathbf{G}^{\mathsf{pke\text{-}cca}}_{\Pi,\mathcal{A}}(\lambda,0)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathbf{G}^{\mathsf{pke\text{-}cca}}_{\Pi,\mathcal{A}}(\lambda,1)\}_{\lambda\in\mathbb{N}}$.

Consider now a modified game $\mathbf{G}^{\mathsf{pke\text{-}rcca}}_{\Pi,\mathcal{A}}(\lambda, b)$ that is identical to the above game, except that step 4. is modified as follows:

4'. $\mathcal{A}$ can ask polynomially many decryption queries. Upon input a query c, the challenger computes $m = \mathsf{Dec}(sk, c)$; if $m \in \{m_0, m_1\}$ return test to $\mathcal{A}$, and otherwise return m.

We say that Π is *replayable* CCA (RCCA) secure if $\{\mathbf{G}^{\mathsf{pke\text{-}rcca}}_{\Pi,\mathcal{A}}(\lambda,0)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathbf{G}^{\mathsf{pke\text{-}rcca}}_{\Pi,\mathcal{A}}(\lambda, 1)\}_{\lambda\in\mathbb{N}}$. Finally, recall that Π is CPA-secure if no decryption query is allowed at all (i.e., step 2. and 4./4'. in the above games are ignored). Prove or disprove:

(a) CPA security $\Rightarrow$ RCCA security.

(b) RCCA security $\Rightarrow$ CPA security.

(c) RCCA security $\Rightarrow$ CCA security.

For each statement, if you think the property holds you need to give an explicit reduction from one notion to the other; on the contrary, if you think the property does not hold, you need to describe a PKE scheme that satisfies one notion but not the other notion.

3. +3-DDH

3 +3-DDH

Let $\mathsf{GroupGen}$ be a PPT algorithm taking as input the security parameter, and outputting the description of a cyclic group $(\mathbb{G}, \cdot)$ of order a prime q, together with a generator $g \in \mathbb{G}$ of the group. Consider the following variant of the standard Decisional Diffie-Hellman (DDH) assumption, dubbed +3-DDH: We say that the +3-DDH assumption holds w.r.t. $\mathsf{GroupGen}$ if for all PPT adversaries $\mathcal{A}$ we have that

$$\left|\Pr\left[\mathcal{A}(\mathsf{params}, g^x, g^y, g^{xy}) = 1\right] - \Pr\left[\mathcal{A}(\mathsf{params}, g^x, g^y, g^{xy+3}) = 1\right]\right| \in \mathbf{negl}(\lambda),$$

where the probabilities are over the random coins of $\mathcal{A}$, and over the choice of $\mathsf{params} = (\mathbb{G}, g, q) \leftarrow\!\!\$\ \mathsf{GroupGen}(1^\lambda)$, and $x, y \leftarrow\!\!\$\ \mathbb{Z}_q^\star$. Prove that DDH implies +3-DDH.

- Febbraio
  1. Hash Functions Combiners

1 Hash Functions Combiners 10 Points

Let $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3$ be three families of hash functions, where

$$\forall i \in [3]: \ \mathcal{H}_i = \{H_s^i : \{0,1\}^* \to \{0,1\}^{\ell_i}\}_{s \in \{0,1\}^\lambda}.$$

You know that at least one of $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3$ is collision resistant, but you don't know which one.

(a) Show how to design a collision-resistant family $\mathcal{H}^* = \{H_s^* : \{0,1\}^* \to \{0,1\}^{\ell^*}\}_{s \in \{0,1\}^{3\lambda}}$, for some $\ell^* \in \mathbb{N}$ to be determined, by combining $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3$.

(b) Does your construction also work for pre-image resistance (i.e. assuming at least one of $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3$ is one-way, is $\mathcal{H}^*$ also one-way)?

2. Multi-Message CPA Security

2 Multi-Message CPA Security 10 Points

Let $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme for variable-length messages. Consider a generalization of CPA security—dubbed *multi-message CPA security*—where the adversary (after seeing the public key) can specify two vectors of messages of size $q \in \mathsf{poly}(\lambda)$, say $\vec{m}_0 = (m_0^1, \ldots, m_0^q)$ and $\vec{m}_1 = (m_1^1, \ldots, m_1^q)$, where $|m_0^i| = |m_1^i|$ for all $i \in [q]$. The challenge ciphertext $\vec{c} = (c_1, \ldots, c_q)$ consists of the component-wise encryption of one of the two vectors.

Formalize the above notion using the indistinguishability paradigm. Hence, prove that standard CPA security implies multi-message CPA security.

3. Weak PRFs

3 Weak PRFs 10 Points

A family of functions $\mathcal{F} = \{F_k : \mathcal{X} \to \mathcal{Y}\}_{k \in \mathcal{K}}$ is called a weak pseudorandom function (wPRF) family if it is like a standard PRF family, except that in the security definition the distinguisher is not allowed to choose the evaluation points $x \in \mathcal{X}$, but rather those are chosen *uniformly at random* by the challenger, and later given to the distinguisher together with the corresponding outputs $y \in \mathcal{Y}$ (either real or random).

(a) Formalize the above definition and show that any PRF family is also a wPRF family.

(b) Show that there is a family $\mathcal{F}^*$ that is weakly pseudorandom but not pseudorandom.

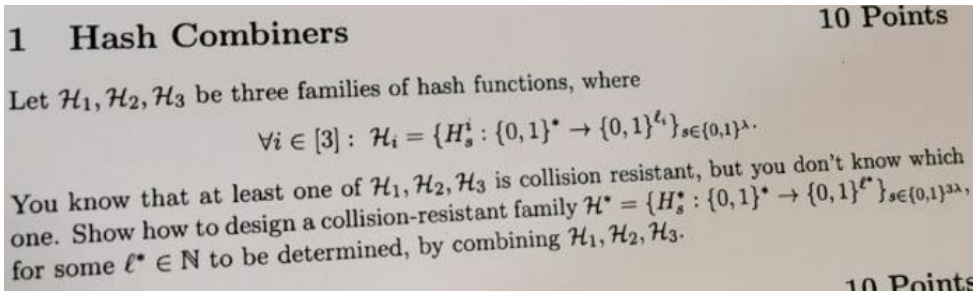
(c) Let $\mathsf{GroupGen}$ be a PPT algorithm taking as input the security parameter, and outputting the description of a cyclic group $(\mathbb{G}, \cdot)$ of prime order q, together with a generator $g \in \mathbb{G}$ of the group. Consider the PRF family $\mathcal{F}_{\mathsf{params}} = \{F_k : \mathbb{G} \to \mathbb{G}\}_{k \in \mathbb{Z}_q}$, with hard-coded parameters $\mathsf{params} = (\mathbb{G}, g, q) \leftarrow\!\!\$\ \mathsf{GroupGen}(1^\lambda)$, and specified by

$$F_k(h) = h^k \in \mathbb{G}.$$

Prove that $\mathcal{F}$ is weakly pseudorandom under the DDH assumption.

2020

- Gennaio
  1. Hash combiners



1 Hash Combiners 10 Points

Let $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3$ be three families of hash functions, where

$$\forall i \in [3]: \ \mathcal{H}_i = \{H_s^i : \{0,1\}^* \to \{0,1\}^{\ell_i}\}_{s \in \{0,1\}^\lambda}.$$

You know that at least one of $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3$ is collision resistant, but you don't know which one. Show how to design a collision-resistant family $\mathcal{H}^* = \{H_s^* : \{0,1\}^* \to \{0,1\}^{\ell^*}\}_{s \in \{0,1\}^{3\lambda}}$, for some $\ell^* \in \mathbb{N}$ to be determined, by combining $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3$.

10 Points

  2. Weak PRFs

2 Weak PRFs

A family of functions $\mathcal{F} = \{F_k : \mathcal{X} \to \mathcal{Y}\}_{k \in \mathcal{K}}$ is called a weak pseudorandom function (wPRF) family if it is like a standard PRF family, except that in the security definition the distinguisher is not allowed to choose the evaluation points $x \in \mathcal{X}$, but rather those are chosen *uniformly at random* by the challenger, and later given to the distinguisher together with the corresponding outputs $y \in \mathcal{Y}$ (either real or random).

(a) Formalize the above definition and show that any PRF family is also a wPRF family.

(b) Show that there is a family $\mathcal{F}^\star$ that is weakly pseudorandom but not pseudorandom.

(c) Let GroupGen be a PPT algorithm taking as input the security parameter, and outputting the description of a cyclic group $(\mathbb{G}, \cdot)$ of prime order q, together with a generator $g \in \mathbb{G}$ of the group. Consider the PRF family $\mathcal{F}_{\mathsf{params}} = \{F_k : \mathbb{G} \to \mathbb{G}\}_{k \in \mathbb{Z}_q}$, with hard-coded parameters $\mathsf{params} = (\mathbb{G}, g, q) \leftarrow_\$ \mathsf{GroupGen}(1^\lambda)$, and specified by

$$F_k(h) = h^k \in \mathbb{G}.$$

Prove that $\mathcal{F}$ is weakly pseudorandom under the DDH assumption.

3. Selective Unforgeability

3 Selective Unforgeability — 10 Points

A signature scheme $\Pi = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ is called selectively unforgeable against chosen-message attacks (SUF-CMA) if no efficient attacker can forge a signature on a message $m^\star$ that is chosen before seeing the public key, even if afterwards (i.e., after learning the public key) the adversary can observe signatures on messages $m \neq m^\star$ of its choice.

Formalize the above notion, and show that it is equivalent to universal unforgeability against chosen-message attacks (UF-CMA) as long as the message space $\mathcal{M}$ satisfies $|\mathcal{M}| \in O(\log \lambda)$, where $\lambda \in \mathbb{N}$ is the security parameter.

- Febbraio
  1. PRFs Combiners

1 PRFs Combiners — 10 Points

For each $i \in [3]$, let $\mathcal{F}_i = \{F_k^i : \{0,1\}^n \to \{0,1\}^n\}_{k \in \{0,1\}^\lambda}$ be a family of functions (i.e., one function for each choice of the secret key). You know that at least one of $\mathcal{F}_1$, $\mathcal{F}_2$, $\mathcal{F}_3$ is a PRF family, but you don't know which one.

Show how to design a PRF family $\mathcal{F}^\star = \{F_{k^\star}^\star \{0,1\}^n \to \{0,1\}^n\}_{k^\star \in \{0,1\}^{3\lambda}}$ by combining $\mathcal{F}_1$, $\mathcal{F}_2$, $\mathcal{F}_3$. (Note that the length of the secret key $k^\star$ is 3λ bits.)

  2. Replayable CCA security

2 Replayable CCA Security — 10 Points

Let $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption (PKE) scheme. Recall the game $\mathbf{G}^{\mathsf{cca}}_{\Pi,\mathcal{A}}(\lambda, b)$ defining chosen-ciphertext attacks (CCA) security, parameterized by Π, a PPT adversary $\mathcal{A}$, and hidden bit $b \in \{0,1\}$:

1. The challenger picks $(pk, sk) \leftarrow_\$ \mathsf{KGen}(1^\lambda)$, and forwards pk to $\mathcal{A}$.
2. $\mathcal{A}$ can ask poly-many decryption queries. Upon input a query c, the challenger returns $m = \mathsf{Dec}(sk, c)$ to $\mathcal{A}$.
3. $\mathcal{A}$ chooses two messages m_0, m_1 of equal length, and receives $c^\star \leftarrow_\$ \mathsf{Enc}(pk, m_b)$.
4. $\mathcal{A}$ can keep asking decryption queries, provided that these queries are different from the challenge ciphertext $c^\star$.
5. $\mathcal{A}$ outputs a bit b'.

We say that a PKE scheme Π is CCA-secure if $\{\mathbf{G}^{\mathsf{cca}}_{\Pi,\mathcal{A}}(\lambda,0)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathbf{G}^{\mathsf{cca}}_{\Pi,\mathcal{A}}(\lambda,1)\}_{\lambda\in\mathbb{N}}$.

Consider now a modified game $\mathbf{G}^{\mathsf{rcca}}_{\Pi,\mathcal{A}}(\lambda,b)$ that is identical to the above game, except that step 4 is modified as follows:

4'. $\mathcal{A}$ can ask poly-many decryption queries. Upon input a query c, the challenger computes $m = \mathsf{Dec}(sk,c)$; if $m \in \{m_0,m_1\}$ return test to $\mathcal{A}$, and otherwise return m.

We say that Π is replayable CCA (RCCA) secure if $\{\mathbf{G}^{\mathsf{rcca}}_{\Pi,\mathcal{A}}(\lambda,0)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathbf{G}^{\mathsf{rcca}}_{\Pi,\mathcal{A}}(\lambda,1)\}_{\lambda\in\mathbb{N}}$. Finally, recall that Π is CPA-secure if no decryption query is allowed at all (i.e., steps 2,4,4' in the above games are ignored). Prove or disprove:

(a) CPA security $\Rightarrow$ RCCA security.

(b) RCCA security $\Rightarrow$ CPA security.

(c) RCCA security $\Rightarrow$ CCA security.

For each statement, if you think the property holds you need to give an explicit reduction from one notion to the other; on the contrary, if you think the property does not hold, you need to describe a PKE scheme that satisfies one notion but not the other notion.

3. +1-DDH

+1-DDH — 10 Points

Let GroupGen be a PPT algorithm taking as input the security parameter, and outputting the description of a cyclic group $(\mathbb{G},\cdot)$ of prime order q, together with a generator $g \in \mathbb{G}$ of the group. Consider the following variant of the standard Decisional Diffie-Hellman (DDH) assumption, dubbed +1-DDH: We say that the +1-DDH assumption holds w.r.t. GroupGen if for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\nu : \mathbb{N} \to [0,1]$ such that

$$\Pr\left[\mathcal{A}(1^\lambda,\rho,g^x,g^y,g^{xy}) = 1\right] - \Pr\left[\mathcal{A}(1^\lambda,\rho,g^x,g^y,g^{xy+1}) = 1\right] \leq \nu(\lambda),$$

where the probabilities are over the random coins of $\mathcal{A}$, and over the choice of $\rho = (\mathbb{G},g,q) \leftarrow\!\!\$\ \mathsf{GroupGen}(1^\lambda)$, and $x,y \leftarrow\!\!\$\ \mathbb{Z}_q$. Prove that DDH implies +1-DDH.

- Maggio
  1. A PRG candidate

1 A PRG Candidate — 10 Points

Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a one-way permutation, and $G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+\ell}$ be a pseudorandom generator with positive stretch (i.e., $\ell \geq 1$). Analyze the following derived construction of a pseudorandom generator $G'_f : \{0,1\}^\lambda \to \{0,1\}^{2\lambda+\ell}$, where

$$G'(s) := (f(s), G(s)).$$

In case you think the derived construction is not secure, exhibit a concrete attack; otherwise, provide a proof of security.

  2. PKE Combiners (*, No IND-CPA nel 2022)

2 PKE Combiners — 10 Points

Let $\Pi_1 = (\mathsf{KGen}_1,\mathsf{Enc}_1,\mathsf{Dec}_1)$ and $\Pi_2 = (\mathsf{KGen}_2,\mathsf{Enc}_2,\mathsf{Dec}_2)$ be two PKE schemes with the same message space $\mathcal{M} = \{0,1\}^n$. You know that at least one of the two PKE schemes is secure, but you don't know which one. Show how to combine Π_1 and Π_2 into a PKE scheme $\Pi = (\mathsf{KGen},\mathsf{Enc},\mathsf{Dec})$, with message space $\mathcal{M}$, such that Π satisfies CPA security as long as at least one of Π_1 and Π_2 satisfies CPA security.

  3. ID scheme based on RSA

3 ID Scheme based on RSA — 10 Points

Consider the following ID scheme $\Pi = (\mathsf{KGen}, \mathcal{P}, \mathcal{V})$.

- The key generation algorithm first computes parameters (N, e, d) as in the RSA cryptosystem. In particular, $N = p \cdot q$ for sufficiently large primes p, q, and $e \cdot d \equiv 1 \bmod (p-1)(q-1)$ with e a prime number. Hence, it picks $x \leftarrow\!\!\$\ \mathbb{Z}_N^*$, computes $y = x^e \bmod N$, and it returns $pk = (N, e, y)$ and $sk = x$.
- One execution of the ID scheme goes as follows: (1) The prover $\mathcal{P}$ picks $a \leftarrow\!\!\$\ \mathbb{Z}_N^*$, and sends $\alpha = a^e \bmod N$ to $\mathcal{V}$; (2) The verifier $\mathcal{V}$ forwards a random challenge $\beta \leftarrow\!\!\$\ \mathbb{Z}_e$ to $\mathcal{P}$; (3) The prover $\mathcal{P}$ replies with $\gamma = x^\beta \cdot a \bmod N$, where x is taken from the secret key and a is the same value sampled in the first round.
- The verifier $\mathcal{V}$ accepts a transcript $\tau = (\alpha, \beta, \gamma)$ if and only if

$$\gamma^e \cdot y^{-\beta} = \alpha \bmod N.$$

Prove that Π is a canonical ID scheme satisfying completeness, special soundness, and honest-verifier zero knowledge under the RSA assumption.

(**Hint:** To prove special soundness you can use the following fact: Given N, elements $u, v \in \mathbb{Z}_N^*$, and integers e, e' for which it holds that $\gcd(|e|, |e'|) = 1$ and $u^e = v^{e'} \bmod N$, an e-th root of v (modulo N) can be computed in polynomial time.)

2021

- Gennaio
    1. Weak collision resistant

1 Weak Collision Resistance — 10 Points

Let $\mathcal{H} := \{h_s : \{0,1\}^n \to \{0,1\}^\ell\}_{s \in \{0,1\}^\lambda}$ be a family of functions, where both n and ℓ depend on the security parameter $\lambda \in \mathbb{N}$ and typically $n(\lambda) \gg \ell(\lambda)$ for all $\lambda \in \mathbb{N}$. Informally, $\mathcal{H}$ is said to be *weakly* collision resistant if the following guarantee holds: No probabilistic polynomial-time adversary $\mathcal{A}$, not knowing the seed $s \in \{0,1\}^\lambda$, can find a pair (x, x') such that $x \neq x'$ and $h_s(x) = h_s(x')$ (i.e., a collision), even when given oracle access to $h_s(\cdot)$. Answer the following questions:

(a) Turn the above notion into a formal definition and explain the difference between weak collision resistance and collision resistance (as defined in class).

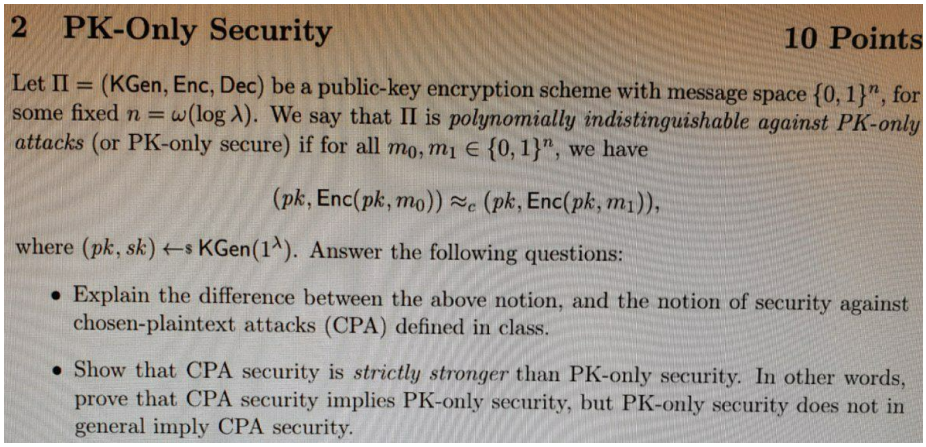
(b) Let Tag be a (deterministic) message authentication code (MAC) with message space $\mathcal{M} = \{0,1\}^\ell$ and key space $\mathcal{K} = \{0,1\}^\lambda$. Consider the following derived (deterministic) MAC Tag' with message space $\mathcal{M}' = \{0,1\}^n$ and key space $\mathcal{K}' = \{0,1\}^{2\lambda}$, based on Tag and on a function family $\mathcal{H}$:

(i) The secret key consists of a pair $k' := (k, s)$, where both $k, s \leftarrow\!\!\$\ \{0,1\}^\lambda$;

(ii) Algorithm Tag' takes the key $k' = (k, s)$, a message $m \in \{0,1\}^n$, and returns $\tau = \mathsf{Tag}(k, h_s(m))$.

Prove that if Tag is UF-CMA and $\mathcal{H}$ is weakly collision resistant, then Tag' is also UF-CMA.

    2. PK-only security



2 PK-Only Security **10 Points**

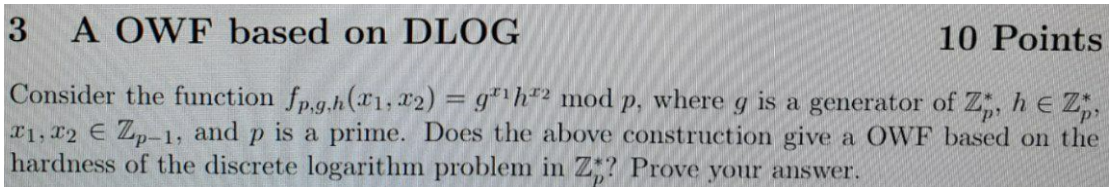
Let $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme with message space $\{0,1\}^n$, for some fixed $n = \omega(\log \lambda)$. We say that Π is *polynomially indistinguishable against PK-only attacks* (or PK-only secure) if for all $m_0, m_1 \in \{0,1\}^n$, we have

$$(pk, \mathsf{Enc}(pk, m_0)) \approx_c (pk, \mathsf{Enc}(pk, m_1)),$$

where $(pk, sk) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$. Answer the following questions:

- Explain the difference between the above notion, and the notion of security against chosen-plaintext attacks (CPA) defined in class.
- Show that CPA security is *strictly stronger* than PK-only security. In other words, prove that CPA security implies PK-only security, but PK-only security does not in general imply CPA security.

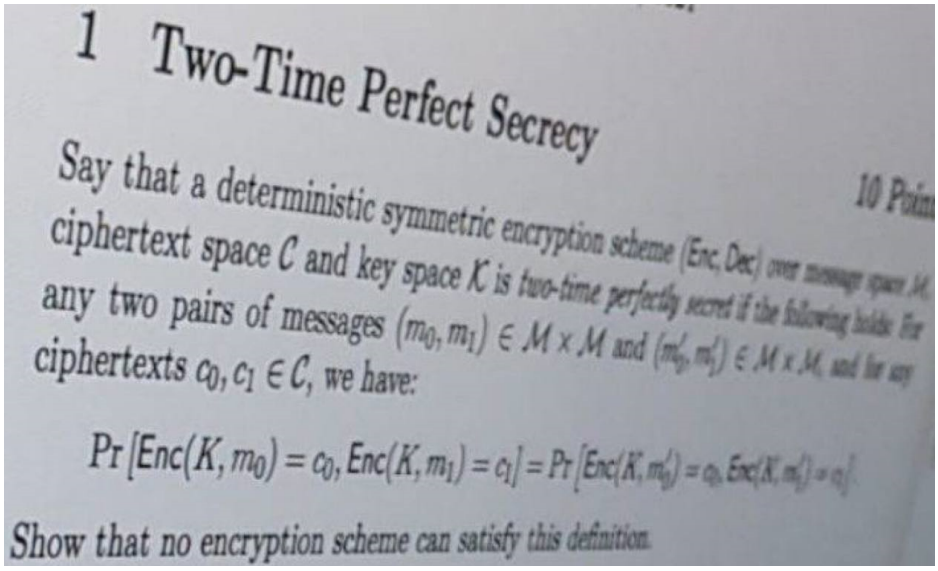
3. A OWF based on DLOG



3 A OWF based on DLOG **10 Points**

Consider the function $f_{p,g,h}(x_1, x_2) = g^{x_1} h^{x_2} \bmod p$, where g is a generator of $\mathbb{Z}_p^*$, $h \in \mathbb{Z}_p^*$, $x_1, x_2 \in \mathbb{Z}_{p-1}$, and p is a prime. Does the above construction give a OWF based on the hardness of the discrete logarithm problem in $\mathbb{Z}_p^*$? Prove your answer.

- Luglio
  1. Two-time Perfect secrecy



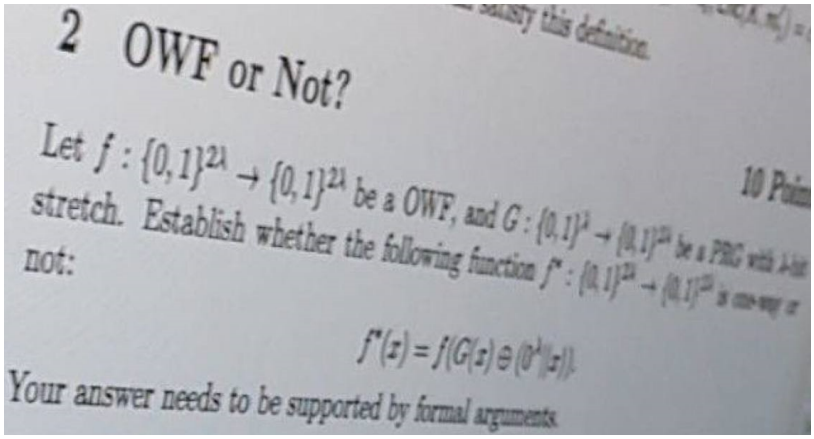
1 Two-Time Perfect Secrecy

Say that a deterministic symmetric encryption scheme (Enc, Dec) over message space $\mathcal{M}$, ciphertext space $\mathcal{C}$ and key space $\mathcal{K}$ is two-time perfectly secret if the following holds for any two pairs of messages $(m_0, m_1) \in \mathcal{M} \times \mathcal{M}$ and $(m_0', m_1') \in \mathcal{M} \times \mathcal{M}$, and for any ciphertexts $c_0, c_1 \in \mathcal{C}$, we have:

$$\Pr[\mathsf{Enc}(K, m_0) = c_0, \mathsf{Enc}(K, m_1) = c_1] = \Pr[\mathsf{Enc}(K, m_0') = c_0, \mathsf{Enc}(K, m_1') = c_1]$$

Show that no encryption scheme can satisfy this definition.

  2. OWF or Not?



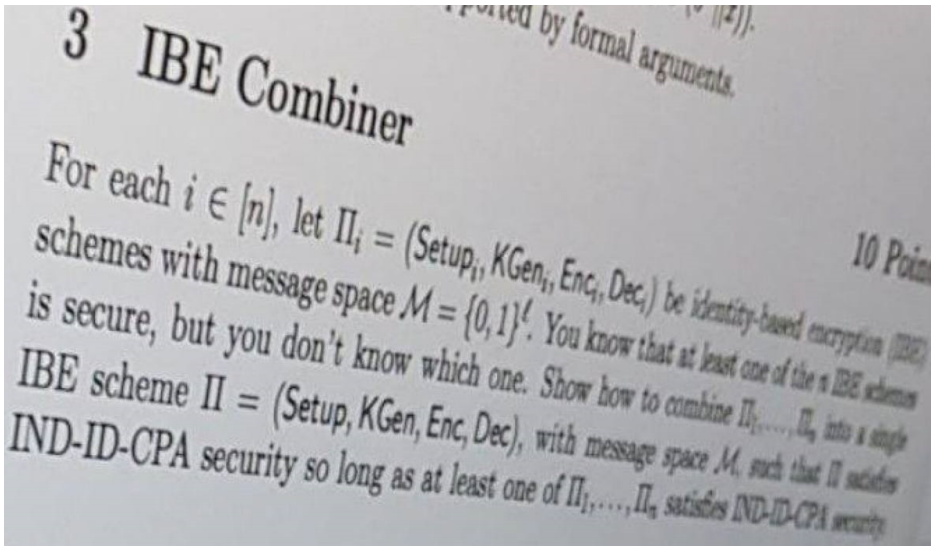
2 OWF or Not?

Let $f : \{0,1\}^{2\lambda} \to \{0,1\}^{2\lambda}$ be a OWF, and $G : \{0,1\}^{\lambda} \to \{0,1\}^{2\lambda}$ be a PRG with λ-bit stretch. Establish whether the following function $f' : \{0,1\}^{2\lambda} \to \{0,1\}^{2\lambda}$ is one-way or not:

$$f'(x) = f(G(x) \oplus (0^{\lambda} \| x))$$

Your answer needs to be supported by formal arguments.

3. IBE Combiner (No IBE nel 2022)



3 IBE Combiner

ted by formal arguments.

For each $i \in [n]$, let $\Pi_i = (\mathsf{Setup}_i, \mathsf{KGen}_i, \mathsf{Enc}_i, \mathsf{Dec}_i)$ be identity-based encryption (IBE) schemes with message space $\mathcal{M} = \{0,1\}^\ell$. You know that at least one of the n IBE schemes is secure, but you don't know which one. Show how to combine $\Pi_1, \ldots, \Pi_n$ into a single IBE scheme $\Pi = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$, with message space $\mathcal{M}$, such that Π satisfies IND-ID-CPA security so long as at least one of $\Pi_1, \ldots, \Pi_n$ satisfies IND-ID-CPA security.

10 Poi

- Settembre
  1. MAC design

1 MAC Design — 10 Points

Let $\mathcal{F} = \{F_k : \{0,1\}^n \to \{0,1\}^n\}_{k \in \{0,1\}^{2\lambda}}$ be a PRF family with keys of length 2λ bits. Assuming one-way functions exist, show how to use $\mathcal{F}$ to construct a secure MAC with key length λ bits and message length $t \cdot n$ bits for any $t \in \mathbb{N}$. Formally prove security of your construction.

  2. Collision Resistant or Not

2 Collision Resistant or Not? — 10 Points

Let $\mathcal{H} = \{H_s : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda\}_{s \in \{0,1\}^\lambda}$ be a CRH family. Consider the modified family $\mathcal{H}^* = \{H_s^* : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda\}_{s \in \{0,1\}^\lambda}$ such that for all $s \in \{0,1\}^\lambda$:

$$H_s^*(x) := \begin{cases} 0^\lambda & \text{if } x = 0^{2\lambda} \\ H_s(x) & \text{otherwise.} \end{cases}$$

Establish whether $\mathcal{H}^*$ still defines a CRH family. (In case you think $\mathcal{H}^*$ is secure, you must prove it formally; otherwise, you need to show a polynomial-time attack breaking collision resistance.)

  3. Discrete Log Attack

3 Discrete Log Attack — 10 Points

Let $\mathbb{G}$ be a cyclic group of order q and with generator g, and assume all group operations can be performed in $\mathbb{G}$ in $O(1)$ time. Let $\mathcal{A}$ be a randomized algorithm running in time T and solving the discrete logarithm problem *on average* with probability .01, meaning that:

$$\Pr[\mathcal{A}(g^x) = x : x \leftarrow\!\!\$\, \mathbb{Z}_q] \geq .01,$$

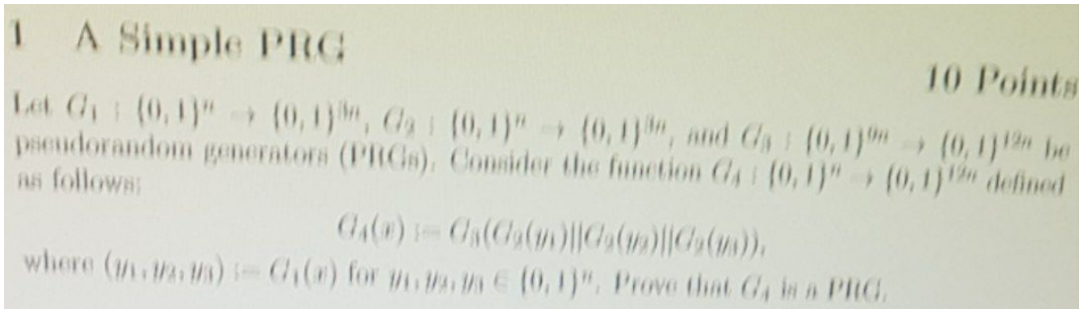
where the probability is taken over a random choice of x and over the randomness of $\mathcal{A}$. Construct a randomized algorithm $\mathcal{B}$ (using $\mathcal{A}$) that runs in time $O(T)$ and solves the discrete logarithm problem in the *worst case* with probability .99, meaning that *for every* $x \in \mathbb{Z}_q$:

$$\Pr[\mathcal{B}(g^x) = x] \geq .99,$$

where the probability is now only over the randomness of $\mathcal{B}$.

2022

- Gennaio
    1. A simple PRG



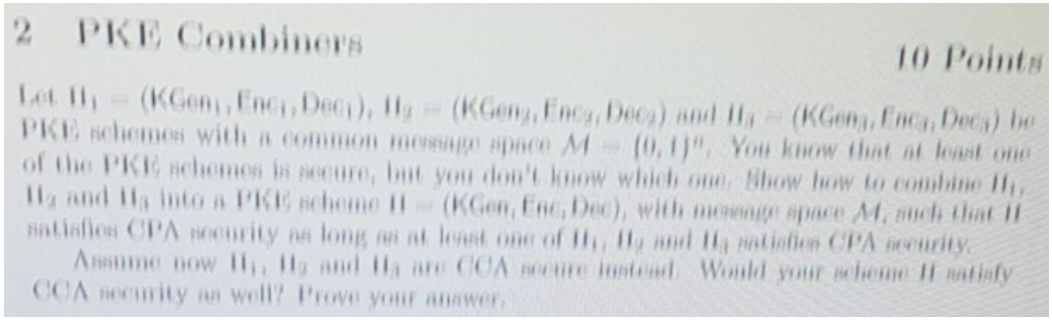
1 A Simple PRG — 10 Points

Let $G_1 : \{0,1\}^n \to \{0,1\}^{3n}$, $G_2 : \{0,1\}^n \to \{0,1\}^{3n}$, and $G_3 : \{0,1\}^{9n} \to \{0,1\}^{12n}$ be pseudorandom generators (PRGs). Consider the function $G_4 : \{0,1\}^n \to \{0,1\}^{12n}$ defined as follows:

$$G_4(x) := G_3(G_2(y_1)\|G_2(y_2)\|G_2(y_3)),$$

where $(y_1, y_2, y_3) := G_1(x)$ for $y_1, y_2, y_3 \in \{0,1\}^n$. Prove that G_4 is a PRG.

    2. PKE combiners

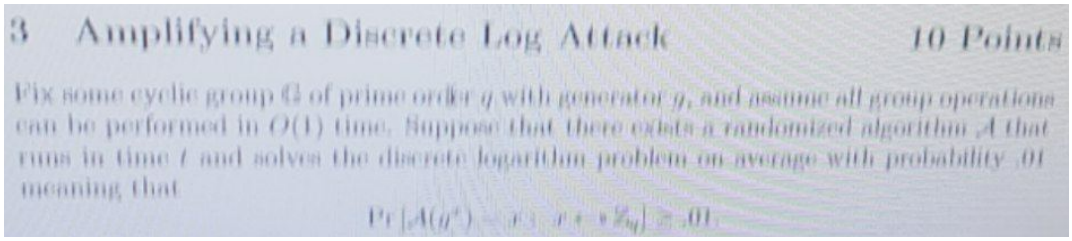


2 PKE Combiners — 10 Points

Let $\Pi_1 = (\mathsf{KGen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$, $\Pi_2 = (\mathsf{KGen}_2, \mathsf{Enc}_2, \mathsf{Dec}_2)$ and $\Pi_3 = (\mathsf{KGen}_3, \mathsf{Enc}_3, \mathsf{Dec}_3)$ be PKE schemes with a common message space $\mathcal{M} = \{0,1\}^n$. You know that at least one of the PKE schemes is secure, but you don't know which one. Show how to combine Π_1, Π_2 and Π_3 into a PKE scheme $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$, with message space $\mathcal{M}$, such that Π satisfies CPA security as long as at least one of Π_1, Π_2 and Π_3 satisfies CPA security.

Assume now Π_1, Π_2 and Π_3 are CCA secure instead. Would your scheme Π satisfy CCA security as well? Prove your answer.

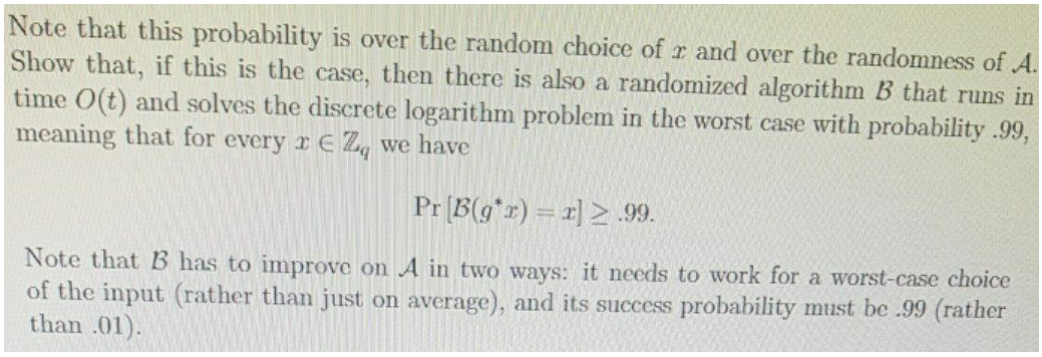
    3. Amplifying a discrete LOG attack



3 Amplifying a Discrete Log Attack — 10 Points

Fix some cyclic group $\mathbb{G}$ of prime order q with generator g, and assume all group operations can be performed in $O(1)$ time. Suppose that there exists a randomized algorithm $\mathcal{A}$ that runs in time t and solves the discrete logarithm problem on average with probability .01 meaning that

$$\Pr[\mathcal{A}(g^x) = x : x \leftarrow \mathbb{Z}_q] \geq .01.$$

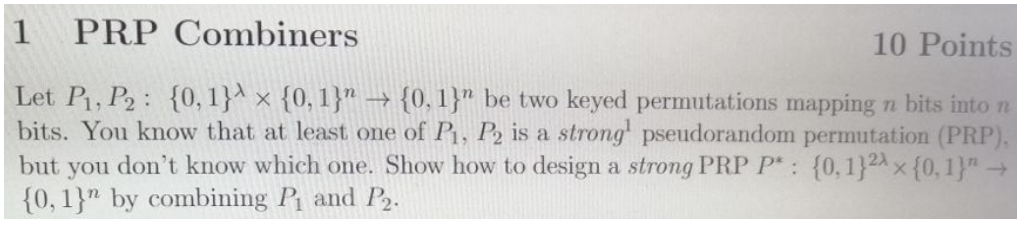


Note that this probability is over the random choice of x and over the randomness of $\mathcal{A}$. Show that, if this is the case, then there is also a randomized algorithm $\mathcal{B}$ that runs in time $O(t)$ and solves the discrete logarithm problem in the worst case with probability .99, meaning that for every $x \in \mathbb{Z}_q$ we have

$$\Pr[\mathcal{B}(g^x) = x] \geq .99.$$

Note that $\mathcal{B}$ has to improve on $\mathcal{A}$ in two ways: it needs to work for a worst-case choice of the input (rather than just on average), and its success probability must be .99 (rather than .01).

- Febbraio
    1. PRP Combiners



1 PRP Combiners — 10 Points

Let $P_1, P_2 : \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^n$ be two keyed permutations mapping n bits into n bits. You know that at least one of P_1, P_2 is a *strong*[1] pseudorandom permutation (PRP), but you don't know which one. Show how to design a *strong* PRP $P^* : \{0,1\}^{2\lambda} \times \{0,1\}^n \to \{0,1\}^n$ by combining P_1 and P_2.

Consider the following ID scheme Π = (

[1]Recall that in the security definition of a *strong* PRP, the adversary is allowed to do both forward and backward evaluation queries to the permutation.

2. Selective Unforgeability

2 Selective Unforgeability — 10 Points

Define a variant of universal unforgeability against chosen-message attacks (UF-CMA) for digital signatures, in which the adversary has to commit to the message $m^* \in \mathcal{M}$ on which he will forge a signature before seeing the public key (where $\mathcal{M}$ is the message space); note that in the definition the adversary should still be allowed to sign arbitrary messages. Name your notion SF-CMA (i.e., selective unforgeability against chosen-message attacks). Prove or disprove:

(a) UF-CMA $\Rightarrow$ SF-CMA.

(b) SF-CMA $\Rightarrow$ UF-CMA.

3. ID Scheme based on RSA

3 ID Scheme based on RSA

Consider the following ID scheme $\Pi = (\mathsf{KGen}, \mathcal{P}, \mathcal{V})$.

- The key generation algorithm first computes parameters (N, e, d) as in the RSA cryptosystem. In particular, $N = p \cdot q$ for sufficiently large primes p, q, and $e \cdot d \equiv 1 \bmod (p-1)(q-1)$ with e a prime number. Hence, it picks $x \leftarrow_{\$} \mathbb{Z}_N^*$, computes $y = x^e \bmod N$, and it returns $pk = (N, e, y)$ and $sk = x$.
- One execution of the ID scheme goes as follows: (1) The prover $\mathcal{P}$ picks $a \leftarrow_{\$} \mathbb{Z}_N^*$, and sends $\alpha = a^e \bmod N$ to $\mathcal{V}$; (2) The verifier $\mathcal{V}$ forwards a random challenge $\beta \leftarrow_{\$} \mathbb{Z}_e$ to $\mathcal{P}$; (3) The prover $\mathcal{P}$ replies with $\gamma = x^\beta \cdot a \bmod N$, where x is taken from the secret key and a is the same value sampled in the first round.
- The verifier $\mathcal{V}$ accepts a transcript $\tau = (\alpha, \beta, \gamma)$ if and only if

$$\gamma^e \cdot y^{-\beta} = \alpha \bmod N.$$

Prove that Π is a canonical ID scheme satisfying completeness and passive security[2] under the RSA assumption.

- Giugno
  1. A simple PRF

1 A Simple PRF — 10 Points

Let m, n be parameters such that $n \gg m$. Let $G_1 : \{0,1\}^\lambda \to \{0,1\}^{3\lambda}$ and $G_2 : \{0,1\}^m \to \{0,1\}^n$ be two pseudorandom generators (PRGs) and let $F_1 : \{0,1\}^{3\lambda} \times \{0,1\}^m \to \{0,1\}^m$ be a pseudorandom function (PRF).

Prove or disprove: The following construction

$$F^*(\kappa, x) = G_2(F_1(G_1(\kappa), x))$$

yields a PRF $F^* : \{0,1\}^\lambda \times \{0,1\}^m \to \{0,1\}^n$.

2. PK-Only Security

2 PK-Only Security 10 Points

Let $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme with message space $\{0,1\}^n$, for some fixed $n = \omega(\log \lambda)$. We say that Π is *polynomially indistinguishable against PK-only attacks* (or PK-only secure) if for all $m_0, m_1 \in \{0,1\}^n$, we have

$$(pk, \mathsf{Enc}(pk, m_0)) \approx_c (pk, \mathsf{Enc}(pk, m_1)),$$

where $(pk, sk) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$. Answer the following questions:

- Explain the difference between the above notion, and the notion of security against chosen-plaintext attacks (CPA) defined in class.
- Show that CPA security is *strictly stronger* than PK-only security. In other words, prove that CPA security implies PK-only security, but PK-only security does not in general imply CPA security.

3. CDH Variants

3 CDH Variants

Consider the following variants of the standard Computational Diffie-Hellman (CDH) problem over a cyclic group $\mathbb{G}$ of prime order q and with generator $g \in \mathbb{G}$:

- **Square Computational Diffie-Hellman (SCDH):** Upon input (params, g^x) for random $x \leftarrow\!\!\$\ \mathbb{Z}_q$ and $\mathsf{params} = (\mathbb{G}, g, q)$, output g^{x^2} in $\mathbb{G}$.
- **Inverse Computational Diffie-Hellman (InvCDH):** Upon input (params, g^x) for random $x \leftarrow\!\!\$\ \mathbb{Z}_q$ and $\mathsf{params} = (\mathbb{G}, g, q)$, output $g^{x^{-1}}$ in $\mathbb{G}$.

Prove that SCDH and InvCDH are equivalent.

- Luglio
  1. Collision Resistant or Not?

1 Collision Resistant or Not? 10 Points

Let $\lambda \in \mathbb{N}$ and $n = n(\lambda) \in \mathbb{N}$ be parameters. Let $\mathcal{H} = \{H_s : \{0,1\}^{2n} \to \{0,1\}^n\}_{s \in \{0,1\}^\lambda}$ be a family of collision-resistant hash functions compressing $2n$ bits into n bits. Furthermore, let $G : \{0,1\}^{n+1} \to \{0,1\}^{2n}$ be a pseudorandom generator stretching $n+1$ bits into $2n$ bits. Prove or disprove: The following construction

$$H^*_s(x) = H_s(G(x))$$

yields a collision-resistant hash function family compressing $n+1$ bits into n bits.

2. Multi-Message CPA Security

2 Multi-Message CPA Security

Let $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme with message space $\mathcal{M}$. Formally define a generalization of CPA security, where the adversary (after seeing the public key) can specify two vectors of messages of size $q \in \mathbb{N}$, say $\vec{m}_0 = (m_0^1, \ldots, m_0^q) \in \mathcal{M}^q$ and $\vec{m}_1 = (m_1^1, \ldots, m_1^q) \in \mathcal{M}^q$, instead of just two messages, and hence it receives the component-wise encryption of one of the two vectors. Name your notion q-message CPA security.

Prove that 1-message CPA security (i.e., the definition of CPA security given in class) implies q-message CPA security, for an arbitrary polynomial $q \in \mathsf{poly}(\lambda)$.

3. A Variant of CDH

3 A Variant of CDH

Consider the following variant of the standard Computational Diffie-Hellman (CDH) problem over a cyclic group $\mathbb{G}$ of prime order q and with generator $g \in \mathbb{G}$:

- **Division Computational Diffie-Hellman (DCDH):** Upon input $(\mathsf{params}, g^x, g^y)$ for random $x, y \leftarrow\!\!\$\ \mathbb{Z}_q$ and $\mathsf{params} = (\mathbb{G}, g, q)$, output $g^{x/y}$ in $\mathbb{G}$.

Prove that DCDH and CDH are equivalent.
(**Hint.** You may assume that CDH is also equivalent to the following problem:

- **Inverse Computational Diffie-Hellman (InvCDH):** Upon input (params, g^x) for random $x \leftarrow\!\!\$\ \mathbb{Z}_q$ and $\mathsf{params} = (\mathbb{G}, g, q)$, output $g^{x^{-1}}$ in $\mathbb{G}$.)

- Settembre
  1. Chain encryption

Chain Encryption **10 Points**

Let $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ be a CPA-secure secret-key encryption scheme over key space $\mathcal{K}$, message space $\mathcal{M}$, and ciphertext space $\mathcal{C}$, such that $\mathcal{K} = \mathcal{M}$. Consider the secret-key encryption scheme $\Pi' = (\mathsf{Enc}', \mathsf{Dec}')$ over key space $\mathcal{K}^2$, message space $\mathcal{M}$, and ciphertext space $\mathcal{C}^2$, where Enc' is defined as follows:

$$\mathsf{Enc}'((k_1, k_2), m) := (\mathsf{Enc}(k_1, k_2), \mathsf{Enc}(k_2, m)).$$

Specify the decryption algorithm Dec', and show that Π' is CPA-secure.

  2. MACs with Multi-key Security

2 MACs with Multi-key Security **10 Points**

Let $\Pi = \mathsf{Tag}$ be a message authentication code over key space $\mathcal{K}$ and message space $\mathcal{M}$. Consider the following generalization of the standard game defining UF-CMA security:

- At the beginning of the game, the adversary outputs a number $t \in \mathbb{N}$ indicating the number of keys it wants to attack; the challenger chooses t random keys $k_1, \ldots, k_t$ from the key space $\mathcal{K}$.
- Upon input a tag query (m, j) from the attacker, where $m \in \mathcal{M}$ and $j \in [t]$, the challenger returns $\mathsf{Tag}(k_j, m)$. These queries can be repeated poly-many times.
- At the end of the game, the attacker returns (j^*, m^*, τ^*) and wins iff $\tau^* = \mathsf{Tag}(k_{j^*}, m^*)$.

Say that Π is t-user UF-CMA if no PPT attacker can win the above game except with negligible probability. Show that every MAC Π that is UF-CMA is also t-user UF-CMA for polynomial t (in the security parameter).

  3. Alternative DDH Characterization

3 Alternative DDH Characterization **10 Points**

Let $\mathbb{G}$ by a cyclic group of prime order q generated by $g \in \mathbb{G}$. Let D be the uniform distribution over $\mathbb{G}^3$. Let D_{dh} be the uniform distribution over the set of all DH-triples $(g^\alpha, g^\beta, g^{\alpha\beta})$. Let D_{ndh} be the uniform distribution over the set of all non-DH-triples $(g^\alpha, g^\beta, g^\gamma)$ with $\gamma \neq \alpha\beta$. Answer the following questions:

- Show that the statistical distance between D and D_{ndh} is $1/q$.
- Show that, under the DDH assumption, the distributions D_{dh} and D_{ndh} are computationally indistinguishable.

- Ottobre
  1. A PRF construction

1 A PRF Construction **10 Points**

Let $\mathcal{F} = \{F_k : \{0,1\}^n \to \{0,1\}^n\}_{k \in \{0,1\}^n}$ be a PRF family. Prove or disprove: The following construction defines a PRF family $\mathcal{F}' = \{F'_k : \{0,1\}^n \to \{0,1\}^n\}_{k \in \{0,1\}^n}$ where

$$\forall k, x \in \{0,1\}^n, F'(k, x) = F(F_k(0^n), x).$$

  2. Circular security

2 Circular Security — 10 Points

Let $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ be a secret-key encryption scheme over key space $\mathcal{K}$, message space $\mathcal{M}$, and ciphertext space $\mathcal{C}$, such that $\mathcal{K} \subseteq \mathcal{M}$. We say that Π is circularly secure if the standard notion of CPA security holds even when the adversary is given $\mathsf{Enc}(k, k)$ where k is the scheme secret key. Give a formal definition of circular security. Prove or disprove: CPA security implies circular security.

3. A signature construction

3 A Signature Construction — 10 Points

Let $\Pi = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a signature scheme over message space $\mathcal{M} = \{0,1\}^n$. Consider the following derived signature scheme $\Pi' = (\mathsf{KGen}, \mathsf{Sign}', \mathsf{Vrfy}')$ over $\mathcal{M}$:

$$\mathsf{Sign}'(sk, m) = (r, \mathsf{Sign}(sk, m \oplus r), \mathsf{Sign}(sk, r)) \text{ for } r \leftarrow\!\!\$\ \{0,1\}^n$$
$$\mathsf{Vrfy}'(pk, m, (r, \sigma_0, \sigma_1)) = (\mathsf{Vrfy}(pk, m \oplus r, \sigma_0)) \wedge (\mathsf{Vrfy}(pk, r, \sigma_1)).$$

Prove or disprove: The above signature scheme Π' is UF-CMA assuming Π is UF-CMA.

2023

- Gennaio
  1. PRG Candidates

1 PRG Candidates

Let $G : \{0,1\}^n \to \{0,1\}^{2n}$ be a PRG. Establish whether the following PRG candidates $G', G'' : \{0,1\}^n \to \{0,1\}^{3n}$ are secure or not:

- $G'(s) = (x \oplus y, u, v)$, where $(x, y) = G(s)$ and $(u, v) = G(y)$;
- $G''(s) = (x, y \oplus u, v)$, where $(x, y) = G(s)$ and $(u, v) = G(y)$.

Prove your answer.

2. CCA Secure Or Not?

2 CCA Secure Or Not?

Consider the following construction of a SKE scheme $\Pi^* = (\mathsf{Enc}^*, \mathsf{Dec}^*)$, based on a PRF family $\mathcal{F} = \{F_k : \{0,1\}^n \to \{0,1\}^n\}_{k \in \{0,1\}^\lambda}$ and on a MAC $\mathsf{Tag} : \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^\lambda$ with UF-CMA security.

Key Generation: The key generation algorithm returns a random key $k^* = (k', k'')$, where $k', k'' \in \{0,1\}^\lambda$.

Encryption: The encryption algorithm takes $k^* = (k', k'')$ and $m \in \{0,1\}^n$ as inputs, and it returns $c^* = (r, c', c'')$ where $r \leftarrow\!\!\$\ \{0,1\}^n$, $c' = F_{k'}(r) \oplus m$, and $c'' = \mathsf{Tag}_{k''}(c')$.

Decryption: The decryption algorithm takes $k^* = (k', k'')$ and $c^* = (r, c', c'')$, and outputs $m = F_{k'}(r) \oplus c'$ if and only if $\mathsf{Tag}_{k''}(c') = c''$ (otherwise, it outputs $\perp$).

Prove or disprove: Π^* achieves CCA security.

3. RSA Variants

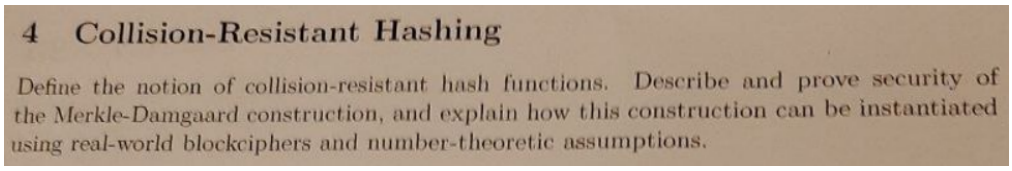
3 RSA Variants

Let RSAGen be the key generation algorithm for the RSA cryptosystem. Consider the following variant of the RSA assumption: We say that the RSA' assumption holds w.r.t. RSAGen if for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\nu : \mathbb{N} \to [0,1]$ such that:

$$\Pr\left[\hat{x} = x : \begin{array}{c}(n, e, d) \leftarrow\!\!\$\ \mathsf{RSAGen}(1^\lambda); x \leftarrow\!\!\$\ \mathbb{Z}_n \\ y = x^e \bmod n; \hat{x} \leftarrow\!\!\$\ \mathcal{A}(n, e, y)\end{array}\right] \leq \nu(\lambda).$$

Note that the only difference with the RSA assumption defined in class is that $x \leftarrow\!\!\$\ \mathbb{Z}_n$ (instead of $x \leftarrow\!\!\$\ \mathbb{Z}_n^*$). Prove that the RSA' assumption holds if and only if the RSA assumption holds.

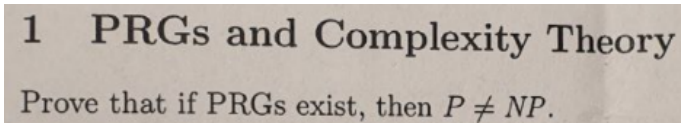
4. (*Theory*) Collision-Resistant hashing



4 Collision-Resistant Hashing

Define the notion of collision-resistant hash functions. Describe and prove security of the Merkle-Damgaard construction, and explain how this construction can be instantiated using real-world blockciphers and number-theoretic assumptions.

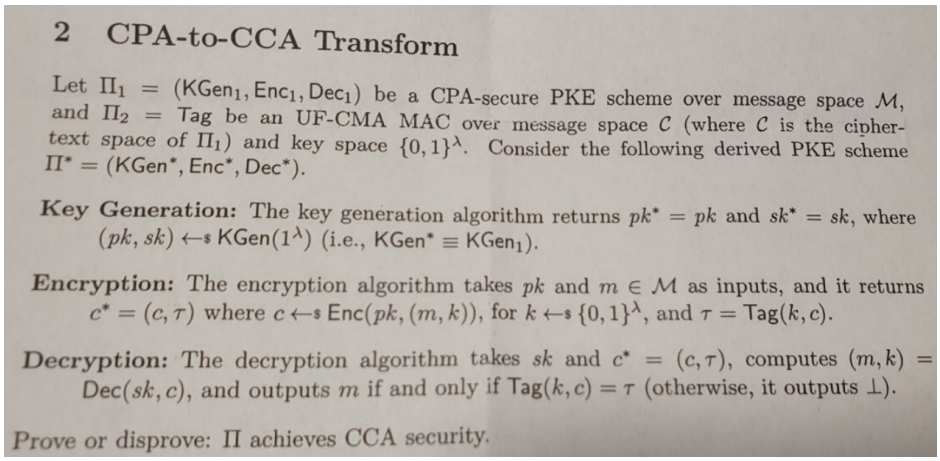
- Febbraio
  1. PRGs and Complexity Theory



1 PRGs and Complexity Theory

Prove that if PRGs exist, then $P \neq NP$.

  2. CPA-to-CCA Transform



2 CPA-to-CCA Transform

Let $\Pi_1 = (\mathsf{KGen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ be a CPA-secure PKE scheme over message space $\mathcal{M}$, and $\Pi_2 = \mathsf{Tag}$ be an UF-CMA MAC over message space $\mathcal{C}$ (where $\mathcal{C}$ is the ciphertext space of Π_1) and key space $\{0,1\}^\lambda$. Consider the following derived PKE scheme $\Pi^* = (\mathsf{KGen}^*, \mathsf{Enc}^*, \mathsf{Dec}^*)$.

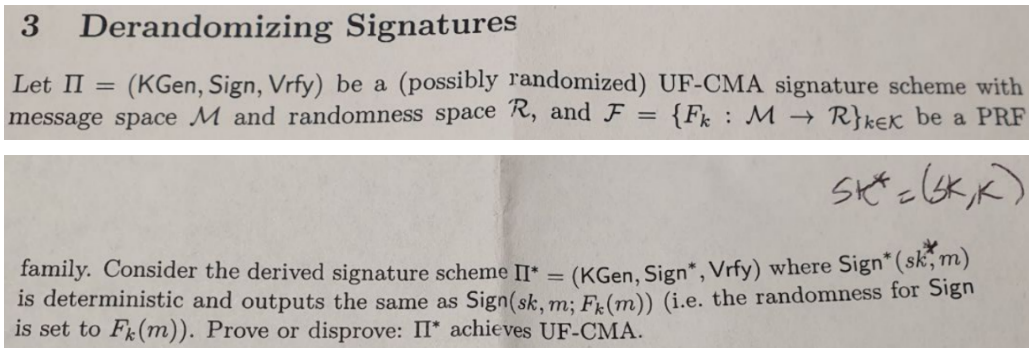
Key Generation: The key generation algorithm returns $pk^* = pk$ and $sk^* = sk$, where $(pk, sk) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$ (i.e., $\mathsf{KGen}^* \equiv \mathsf{KGen}_1$).

Encryption: The encryption algorithm takes pk and $m \in \mathcal{M}$ as inputs, and it returns $c^* = (c, \tau)$ where $c \leftarrow\!\!\$\ \mathsf{Enc}(pk, (m, k))$, for $k \leftarrow\!\!\$\ \{0,1\}^\lambda$, and $\tau = \mathsf{Tag}(k, c)$.

Decryption: The decryption algorithm takes sk and $c^* = (c, \tau)$, computes $(m, k) = \mathsf{Dec}(sk, c)$, and outputs m if and only if $\mathsf{Tag}(k, c) = \tau$ (otherwise, it outputs $\perp$).

Prove or disprove: Π achieves CCA security.

  3. Derandomizing Signatures



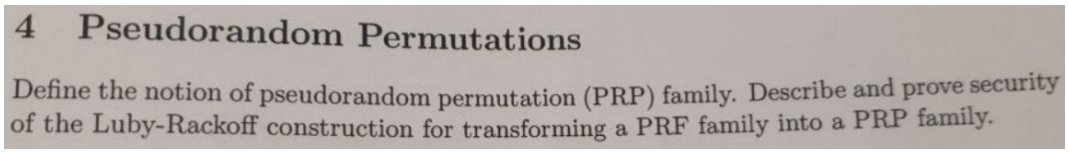
3 Derandomizing Signatures

Let $\Pi = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a (possibly randomized) UF-CMA signature scheme with message space $\mathcal{M}$ and randomness space $\mathcal{R}$, and $\mathcal{F} = \{F_k : \mathcal{M} \to \mathcal{R}\}_{k \in \mathcal{K}}$ be a PRF

$SK^* = (SK, K)$

family. Consider the derived signature scheme $\Pi^* = (\mathsf{KGen}, \mathsf{Sign}^*, \mathsf{Vrfy})$ where $\mathsf{Sign}^*(sk^*, m)$ is deterministic and outputs the same as $\mathsf{Sign}(sk, m; F_k(m))$ (i.e. the randomness for Sign is set to $F_k(m)$). Prove or disprove: Π^* achieves UF-CMA.

  4. (*Theory*) Pseudorandom Permutations



4 Pseudorandom Permutations

Define the notion of pseudorandom permutation (PRP) family. Describe and prove security of the Luby-Rackoff construction for transforming a PRF family into a PRP family.

Un ringraziamento ❤ per Anna, Elisa, Michela, Betta e Martina.