UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI
FACULTATEA DE INFORMATICĂ

Lucrare de licență

Generarea, derivarea, schimbul și managementul cheilor criptografice

propusă de Alina-Elena BRÎNZĂ

Sesiunea: Iulie, 2017
Coordonator științific: Ferucio Laurențiu Țiplea

UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI
FACULTATEA DE INFORMATICĂ

Generarea, derivarea, schimbul și managementul cheilor criptografice

Alina-Elena BRÎNZĂ

Sesiunea: Iulie, 2017
Coordonator științific: Ferucio Laurențiu Țiplea

DECLARAȚII PRIVIND ORIGINALITATEA ȘI RESPECTAREA DREPTURILOR DE AUTOR

Prin prezenta declar că Lucrearea de licență cu titlul "Generarea, derivarea, schimbul și managementul cheilor criptografice" este scrisă de mine și nu a mai fost prezentată niciodată la o altă facultate sau instituție de învățământ superior din țară sau din străinatate. De asemenea, declar că toate sursele utilizate, inclusiv cele preluate de pe Internet, sunt indicate în lucrare, cu respectarea regulilor de evitare a plagiatului:

- toate fragmentele de text reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și dețin referința precisă a sursei;
- reformularea în cuvinte proprii a textelor scrise de către alți autori deține referința precisă;
- codul sursa, imaginile etc. preluate din proiecte open-source sau alte surse sunt utilizate cu respectarea drepturilor de autor și dețin referințe precise;
- rezumarea ideilor altor autori precizează referința precisă la textul original.

Iași, 03.07.2017

Absolvent Alina-Elena BRÎNZĂ

_____

(semnătura în original)

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul "Generarea, derivarea, schimbul și managementul cheilor criptografice", codul sursa al programelor și celelalte continuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea "Alexandru Ioan Cuza" din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași, 03.07.2017

Absolvent Alina-Elena BRÎNZĂ

_____

(semnătura în original)

ACORD PRIVIND PROPRIETATEA DREPTULUI DE AUTOR

Facultatea de Informatică este de acord ca drepturile de autor asupra programelor-calculator, în format executabil și sursă, să aparțină autorului prezentei lucrări, Alina-Elena BRÎNZĂ.

Încheierea acestui acord este necesară din urmatoarele motive:

- Au fost utilizate materiale științifice precum: articole științifice, demonstrații, tutoriale, cercetări, imagini care au fundamentat din punct de vedere științific prezenta lucrare de licență.

Iași, 03.07.2017

Absolvent Alina-Elena BRÎNZĂ

_____

(semnătura în original)

Decan, Prof. Dr. Adrian Iftene

_____

(semnătura în original)

Introducere

Alegerea temei de licenţă "Generarea, derivarea, schimbul şi managementul cheilor criptografice" reprezintă dorinţa de a studia în corelaţie domeniile "Criptografie" şi "Securitatea informaţiei".

Aprofundarea acestor materii reprezintă o introducere tehnică mai amplă în acest domeniu, ce se află într-o continuă dezvoltare. De asemenea, dorinţa de a cerceta algoritmi criptografici deja existenţi cautând noi atacuri sau îmbunătaţiri, studierea metodelor şi tehnicilor de securitate într-o lume în care noi atacuri cibernetice sunt testate zilnic (WannaCry, Petya, Zepto etc.) a semnificat o dezvoltare personală şi profesională.

Ţin să multumesc domnului profesor Ferucio-Laurenţiu Ţiplea pentru întreaga coordonare, sprijinul şi încurajarea oferită atât pe parcursul anilor de studiu cât şi în demersul elaborării lucrării de licenţă.

Mulţumiri adresez şi întregului colectiv de profesori cu care am avut deosebita plăcere să colaborez pe parcursul acestor trei ani de studiu, cât şi tuturor colegilor, alături de care am reuşit să elaborez proiecte, teme de laborator dar şi alături de care anii de studenţie au părut să fie unii dintre cei mai frumoşi.

Cuprins

1. Necesitate în contextul protocolului SSL/TLS

1.1 Argumentare

Necesitatea existenței și studierii generării, derivării, schimbului și managementului de chei criptografice este argumentată de asigurarea securității informatice pentru diverse protocoale de securitate, precum:

- SSL - Secure Socket Layer (nivelul transport)
- TLS - Transport Layer Security (nivelul transport)
- IPsec - Internet Protocol cu elemente de securitate (nivelul rețea)
- HTTPS - Hypertext Transfer Protocol cu elemente de securitate (nivelul aplicație)
- PGP/S-MIME - Pretty Good Privacy / Multiple Internet Mail Extensions cu elemente de securitate
- SSH - Secure Shell

Aceste protocoale de securitate au ca scop asigurarea unei palete largi de prorietăți cum ar fi:

- Integritate: protejarea informațiilor de posibile modificări pe care unele părți neautorizate le pot aduce
- Confidențialitate: protejarea datelor de expunerea la părți neautorizate
- Autenticitate: asigurarea accesului părților autorizate la informații
- Controlul accesului: acces restrictiv bazat pe reguli, pentru diferite părți ale sistemului
- Securitatea în rețea: măsuri hardware și software pentru protecția rețelei
- Managementul securității: generare, stocare, schimb de chei și securitate între părți
- Disaster recovery (Recuperare de date) : recuperarea datelor pierdute în caz de atac

Pentru a putea discuta despre generarea cheilor criptografice, este necesară o scurtă introducere în criptografie.

1.2 Inițiere în termeni criptografici și de securitate

În acest scurt rezumat a ceea ce definește domeniul criptografic, ne propunem să introducem cele mai importante noțiuni, ce vor fi amintite ulterior, cat și să reprezentăm pe o axă a timpului, evoluția și numele marcante în domeniu.

1.2.1 Sistem de criptare / Criptosistem

Fie M o mulțime finită de mesaje (plaintexte), $\mid M \mid > 1$, o mulțime M al cărei cardinal este mai mare ca 1. (mulțimea conține cel puțin un mesaj plaintext).

Reprezentăm un sistem de criptare peste M un 3-uplu $S=(G, E, D)$ unde:

1. G este un generator random de chei de criptare (K reprezintă mulțimea cheilor de criptare $k1, k2, k3, .., kn$)
2. E este un algoritm probabilist (executat de mai multe ori peste același mesaj m, output-ul c poate fi diferit) care pornind de la o cheie k și un mesaj m, generează un criptotext c = E(k,m).
3. D este un algoritm determinist de decriptare care, pornind de la un criptotext c ϵ C , aparținând mulțimii de criptotexte C, și o cheie k ϵ K, aparținând mulțimii de chei K generează / descoperă mesajul inițial m = D(k,c) aparținând mulțimii de mesaje M

Observatie 1. *Orice schemă trebuie să satisfacă proprietatea de corectitudine.*

Observatie 2. *Orice sistem de criptare este caracterizat prin trei distribuții de probabilitate peste spațiile M (mulțimea mesajelor), K (mulțimea cheilor de criptare) și C (mulțimea criptotextelor):*

- O distribuție de probabilitate peste M: indusă natural de alegerea mesajelor care se criptează. P(m) reprezintă probabilitatea cu care este ales mesajul m pentru a putea fi criptat.
- O distribuție de probabilitate peste K: P(k) reprezintă probabilitatea cu care este generată cheia k pentru a putea fi utilizată la criptare.
- O distribuție de probabilitate peste C: indusă de distribuțiile explicate anterior cât și de algoritmul de criptare E.

Notăm cu P(c/m) probabilitatea ca un criptotext c ϵ C să fie obținut cu algoritmul de criptare care primește la intrare mesajul m ϵ M.

P(c) = $\sum_m P(c/m) \cdot P(m)$ adică probabilitatea unui criptotext c ϵ C este suma

probabilităților ca acel criptotext c să fie obținut cu algoritmul de criptare care primește la intrare mesajul m ϵ *M* înmulțit cu probabilitatea ca m ϵ *M* să fie acel mesaj inițial.

P(m,k) = P(m) $\cdot$ P(k), unde P(m) si P(k) sunt distribuții independente, $\forall$ k, c, m, P(k) > 0, P(c) > 0, P(m) > 0 reprezintă probabilitatea ca un mesaj m ϵ *M* să fie criptat cu cheia k ϵ *K*.

1.2.2 Securitate

În linii mari, securitatea se discută prin intermediul a doi factori: indistingibilitatea față de adversar și non-maleabilitatea.

1. Indistingibilitate față de adversar: Date două mesaje pe care adversarul le cunoaște, și primind criptotextul unuia din cele două mesaje, adversarul nu poate distinge cu probabilitate ne-neglijabilă de la care mesaj provine criptotextul.

2. Non-maleabilitate: Dat un criptotext al unui mesaj, ambele cunoscute de adversar, acesta nu poate transforma criptotextul într-un alt criptotext corespunzător unui mesaj înrudit cu mesajul inițial cu probabilitate ne-neglijabilă.

1.2.3 Criptografie simetrică

Un sistem de criptare simetrică este un 3-uplu de forma (*G,E,D*) notat cu S. Elementele *G, E, D* sunt descrise mai jos:

- *G* este un algoritm probabilist de complexitate timp polinomială care, pornind de la parametrul de securitate λ, generează o cheie k ϵ *K*; k $< -G(\lambda)$.

- *E* este un algoritm probabilist de complexitate timp polinomială care, pornind de la o cheie k ϵ *K*, și un mesaj m ϵ *M*, generează un criptotext c ϵ *C*, c $< -E(k, m)$.

- *D* este un algoritm determinist de complexitate timp polinomială care, pornind de la o cheie k ϵ *K* și un criptotext c ϵ *C*, generează un mesaj m ϵ *M*, m $< -D(k, c)$.

Observatie 3. *Pentru orice cheie k ϵ K generată de algoritmul G, pentru orice mesaj m ϵ M, criptotextul c ϵ C generat de algoritmul de criptare E, c $\epsilon E(k, m)$, mesajul m ϵ M obținut prin aplicarea algoritmului D asupra cheii k și criptotextului c, m=D(k, c) și λ=parametru de securitate, mulțimea cheilor K este mulțimea funcției f aplicată peste λ, | K | = f(λ), mulțimea mesajelor M este dată de mulțimea { 0,1 }$^{l(}\lambda$), unde l este un polinom iar l(λ) $\geq \lambda$.*

1.2.4 Criptografie asimetrică / cu chei publice

O schemă de criptare cu chei publice este un sistem *S* = (*G,E,D*) unde elementele *G,E,D* sunt descrise mai jos:

- G este un un algoritm probabilist de complexitate timp polonimială în care $G(1^\lambda)$ generează perechi de chei (cheie publică, cheie secretă) (pk, sk).
- E reprezintă algoritmul de criptare prin care se obține criptotextul c ϵ C folosind cheia publică generată anterior de către generatorul G, c $< -E(pk, m)$.
- D este un algoritm determinist de complexitate timp polinomială care, aplicat criptotextului c ϵ C împreună cu cheia secretă sk, descoperă mesajul inițial criptat m $< -D(sk, c)$

Determinarea cheii secrete sk pornind de la cheia publică pk și mulțimea $\{ 1^\lambda \}$ devine o problemă intractabilă. De asemenea, cheia publică pk poate fi facută public fără a afecta în vreun fel securitatea.

1.2.5 Securitate semantică

Dacă schema de securitate S asigură indistingibilitate față de adversarul pasiv, putem afirma că niciun algoritm probabilist nu poate decide cu probabilitate neneglijabilă o proprietate calculabilă în timp polinomial asupra mesajului de la care provine criptotextul, introducând astfel noțiunea de "securitate semantică".

1.2.6 Securitate IND-CPA

Presupunând că avem un adversar activ în sistemul nostru de securitate, putem distinge două tipuri de atacuri posibile descrise mai jos:

1. CPA (Chosen Plaintext Attack) : adversarul poate obține criptarea pentru un plaintext / un număr de plaintexte alese de el.
2. CCA (Chosen Ciphertext Attack) : adversar ce înglobează un adversar CPA, iar în plus poate obține plaintextul unor criptotexte alese de el.

Securitatea față de aceste atacuri se mai numește securitate IND-CPA și respectiv IND-CCA.

IND-CPA Encryption	Enc (ms)	Dec (ms)	Cpxt (bytes)
Lizard	**0.010**	**0.020**	**745**
CHK+	0.314	0.106	770
IND-CCA Encryption	Enc (ms)	Dec (ms)	Cpxt (bytes)
CCA-Lizard	**0.012**	**0.025**	**778**
CCA-CHK+	0.313	0.302	804

1.2.7 Construcție scheme IND-CPA

Schemele IND-CPA se pot construi în două modalități. Fie prin PRF (Functii Pseudo-random) fie prin PRG (Generatori pseudo-random).

GENERATORI PSEUDO-RANDOM:

Un generator pseudo-random este un generator pentru care distribuțtia de probabilitate a elementelor generate de el este indistingibilă față de distribuția de probabilitate uniforma pe mulțimea elementelor posibile generate de generator (diferențele de probabilitate sunt neglijabile).

FUNCȚII PSEUDO-RADNOM:

O funcție pseudo-random este o familie de funcții indistingibilă față de familia tuturor funcțiilor. Un adversar care primeste o funcție nu va putea distinge cu probabilitate ne-neglijabilă daca funcția este aleasa din acea familie sau dacă este o funcție arbitrară aleasă random. In criptografie notăm funcțiile pseudo-random prin F = (Fk) unde k este în mod uzual o cheie. Alegerea unei funcții din F se reduce la generarea unei chei criptografice k.

Așadar,putem afirma că o schemă de criptare simetrică S este IND-CPA dacă este sigură că pentru orice algoritm A avem adevarată urmatoarea afirmație:

Probabilitatea ca un adversar să câștige experimentul este neglijabilă.

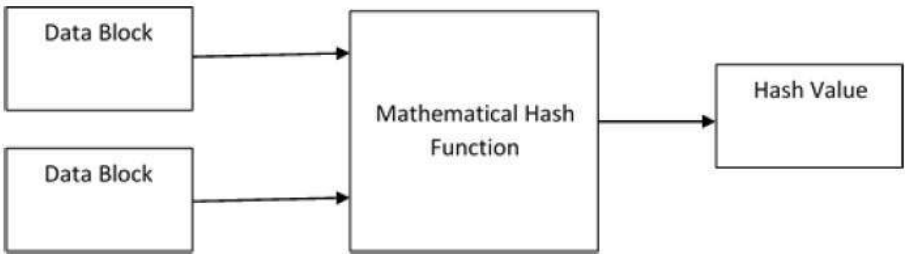
$\mid P(Priv^{CPA}_{A,S}(\lambda) = 1) - 1/2 \mid$ este functie neglijabilă în λ unde $Priv^{CPA}_{A,S}(\lambda)$ este experimentul pentru adversarul A.

Exemple de scheme:

- OTP+PRF
- ECB (Electronic Chainblock)
- CBC (Chaining Block)
- OFB (Offset Block)
- CTR (counter)
- CBC-MAC (CBC cu Message Authenticating Code)
- XCBC-MAC (XCBC cu Message Authenticating Code)
- Nested-MAC (Nested Message Authenticating Code)

1.2.8 Funcții hash

O funcție hash este o funcție al cărei domeniu este $\{0,1\}^*$ și codomeniul este $\{0,1\}^l$. $f : \{0,1\}^* -> \{0,1\}^l$ unde l ≥ 1 este fixat.



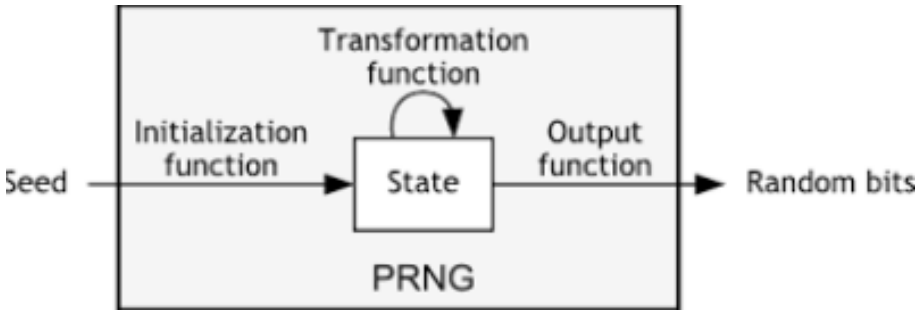
Proprietăți ale funcțiilor hash:

Este de dorit ca funcțiile hash sa aibă anumite proprietăți precum:

1. Rezistența la coliziuni:
   - Pentru un posibil atacator A este intractabil a determina două mesaje diferite și de lungimi diferite $m1, m2 \; \epsilon \; \{0,1\}^*$, $m1 \neq m2$ dacă funcția hash aplicată primului mesaj este identică cu funcția hash aplicată celui de-al doilea mesaj: h($m1$)=h($m2$), ($m1$, $m2$) însemnând o coliziune a funcției hash.
   - Unui atacator A, având mesajul $m1$ îi este intractabil a determina mesajul $m2 \neq m1$ astfel încât $(m1, m2) =$ să fie o coliziune.
2. Funcții one-way:
   - Funcția hash f folosită este o funcție ușor de calculat însă foarte greu de inversat. Așadar, dat y $\epsilon \; 0,1^l$ atacatorului îi este intractabil a determina x, f(x)=y.

1.2.9 Număr random

Domeniul Critpografiei și al Securității informatice, folosesc frecvent conceptul de numar random. (number once used). Extinzând această afirmație, putem spune că o secvență random este o secvență de numere random, unic utilizată.



Având un șir de numere random generate $a_l, a_2, ..., a_k$, numărul random a_{k+1} este impredictibil în raport cu secvența generată anterior.

> Un număr este random dacă are proprietăți statistice "bune" (test statistic ≥ 0.70) iar o secvență random trebuie să asigure impredictibilitate forward și backward.

1.3 Evoluția securității în timp

Securitatea împreună cu noțiunile criptografice (schimb, generare, derivare, management de chei) s-a impus ca modalitate de prevenire a mai multor aspecte precum:

1. Scurgeri de informații către persoane neautorizate (confidențialitate).
2. Modificare de date de către persoane neautorizate (integritate).
3. Respingerea folosirii informațiilor proprii(denial of use, exemplu: DDOS)(valabilitate).

În axa ce urmează, vom pune accentul atât pe punctele de interes în evoluția securității informatice cât și a criptografiei, începând cu anul 1970 și până în prezent.

Tabela 1.1 Event

1970	Bell La Padula gândește o politică de determinare al controlului accesului pentru domeniul militar.
1970	Rețelele Feistel $\Longrightarrow$ un bloc de mesaje este criptat repetând asupra lui un număr de transformări identice, parametrizate după chei de sesiune.
1970	Se pune accentul pe securitatea computerelor și pe îndeplinirea fiabilității.
1976	Diffie, Hellman, Merkle - aduc o soluție pentru distribuția de chei.
1977	Diffie, Hellman, (Merkle), criptosistem bazat pe problema rucsacului.
1978	DES - primul criptosistem construit pe baza rețelelor Feistel
1978	Rivest, Shamir, Adleman (RSA) - primul sistem de cheie publică construit, luând naștere criptografia cu chei publice, rezolvându-se astfel problema schimbului de chei.
1980	Se trece de la ideea securizării computerelor la securizarea informațiilor stocate pe acestea. Se pune accentul pe confidențialitate și izolarea datelor.
1984	Modelul schematic ce se situeaza între TG (Take Grant) și ACM.
1984	Apare criptografia bazată pe identitate.
1990	Mediul Internetului și includerea calculatoarelor în aceasta într-un mod sigur. Se pune accentul pe integritate și valabilitatea datelor.
1996	Schneier pune accentul pe nepredictibilitate și nereproductibilitate în timp fezabil în ceea ce privește generarea de chei.
1998	Mașină cu mai mult de 5000 procesoare legate în paralel, compusă pentru un atac brute-force asupra DES.
2000	Internetul devine ceva indispensabil. Se pune problema securizării datelor dar de această dată și în rețea. Se pune punctul pe non-repudiere, posesia datelor și autenticitatea datelor.
2001	Criptare bazată pe reziduuri pătratice, criptare bazată pe aplicații biliniare.
2001	Atac asupra RC4.
2002	AES înlocuiește DES care fusese decriptat în 1998 în 22 ore.
2002	Primalitatea poate fi rezolvată polinomial (AKS).
2010	Plus de securitate adus în primul rând datorita exploziei rețelelor sociale: cyber security, transparența, eficientizarea costurilor, eficientizarea securizării.
2014	Atac puternic asupra RC4 (*dezvoltat în capitolul următor)

1.4 SSL/TLS - scurtă prezentare a protocolului

SSL = Secure Socket Layer și TLS = Transport Layer Security sunt protocoale ce asigură securitatea comunicării în rețea, câteva dintre aplicațiile de bază ale acestor protocoale fiind: căutare pe browser, poșta electronică (e-mail), mesagerie instantă etc.

Conexiunea privată este asigurată de criptarea simetrică folosită în criptarea datelor trimise, ajutată de ceea ce numim TLS handshake unde cheile sunt generate la fiecare nouă conexiune, pe baza unui secret partajat, ce asigură securitate față de orice adversar, ce dorește modificarea / coruperea datelor. Orice atacuri pot fi așadar detectate, asigurându-se confidențialitate, integritate și valabilitate, termeni deja discutați anterior în acest capitol.

Pași importanți:

1. Key Exchange Key agreement:

   Înainte de a avea loc transferul de date, părțile cad de comun acord asupra unei chei de criptare k, generată de generatori precum: RSAPub_{K}, RSAPriv_{K}, Diffie Hellman efemer, Diffie Hellman pe curbe eliptice, Diffie Hellman efemer pe curbe eliptice, Diffie Hellman anonim, etc.(TLS_RSA, TLD_DH, TLS_DHE, TLS_ECDH) dintre care doar Diffie Hellman efemer (TLS_DHE) și TLS_ECDH asigură secret forward și nu sunt vulnerabile la atacul Man-In-The-Middle.

2. Cipher:

   Aplicarea unor algoritmi ce prevăd atât criptare cat și decriptare; criptosisteme simetrice sau asimetrice, din care amintim de: AES, DES, CBC, SEED CBC, RC2, CBC pentru block cipher și de RC4 pentru stream cipher.

3. Integritatea datelor:
   Asigurarea integrității datelor se realizează prin mecanisme de autentificare a mesajelor precum:

   - MAC (Message Authentication Code) : folosit pentru integritatea datelor
   - HMAC : folosit pentru CBC
   - AEAD : folosit pentru criptare autentificată

Cum este asigurată integritatea, autenticitatea și confidențialitatea în SSL&TLS?

- În timpul autentificării, atât a clientului cat și a serverului, este necesară criptarea datelor cu una din cheile din perechea de chei simetrice, și decriptate cu cealalta cheie ramasă.

1.4.1 SSL&TLS: Autentificarea

AUTENTIFICARE:

Pentru autentificarea serverului:
Clientul folosește cheia publică Pub_K pentru a cripta informațiile care sunt necesare pentru a calcula cheia secretă $Secret_K$. Serverul poate genera o cheie secretă doar dacă poate să decripteze informațiile cu cheia publică $Priv_K$.

Pentru autentificarea clientului:
Serverul folosește cheia publică Pub_K din certificatul clientului pentru a decripta informațiile pe care clientul le trimite (certificatul). Schimbul de mesaje criptate cu cheia secretă $Secret_K$ (Client "finished", Server "finished"), confirmă că autentificarea a fost completă și cu succes.

Observatie 4.

Dacă chiar și unul din pașii de autentificare de mai sus se termină cu eșec, SSL handshake eșuează și sesiunea se termină.

Observatie 5.

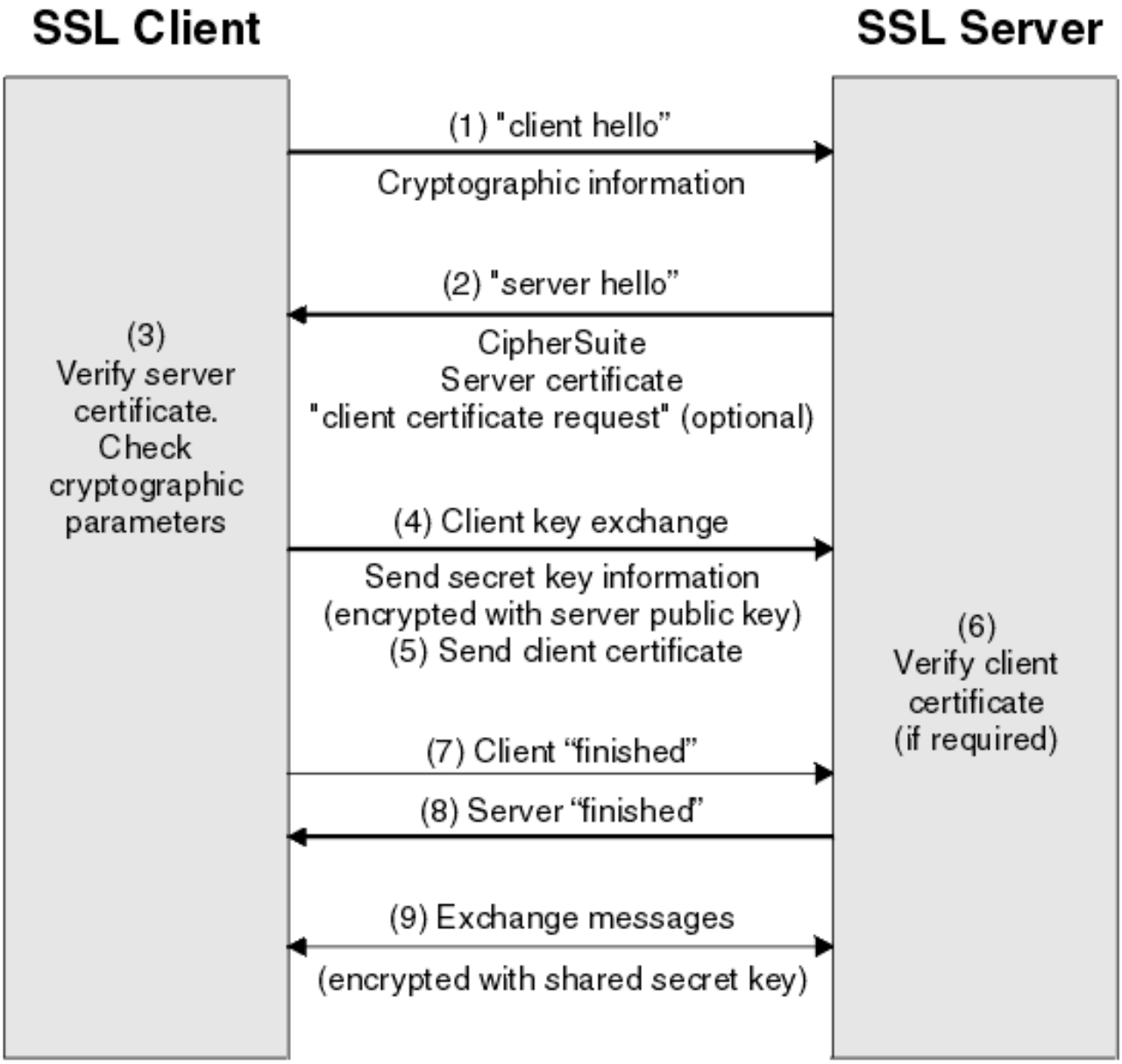
Schimbul de certificate digitale în timpul SSL/TLS handshake este o parte a procesului de autentificare.

1.4.2 SSL&TLS: Confidențialitatea

CONFIDENȚIALITATE:

- SSL&TLS folosește o combinație de criptări simetrice și asimetrice pentru a asigura securitatea mesajelor.
- În timpul handshake-ului, clientul SSL&TLS și serverul, stabilesc un algoritm de criptare A și o cheie secretă $Secret_K$ unică (per sesiune).
- Toate mesajele transmise între clientul SSL&TLS și server sunt criptate folosind algoritmul agreat A, asigurând faptul că mesajele rămân private chiar și în cazul în care sunt interceptate.
- Deoarece SSL&TLS folosește criptare asimetrică la transportul cheii secrete $Secret_K$, nu apar probleme legate de distribuția cheilor criptografice.

1.4.3 SSL Handshake



2. Generarea cheilor

În capitolul anterior am descris pe scurt, la nivelul protocolului SSLTLS, faptul că generarea de chei se realizează pe baza unor algoritmi și generatori.

Capitolul curent este menit să prezinte tehnici de generare a cheilor atât în schemele de criptare simetrică cât și în schemele de criptare asimetrică.
Pentru a argumenta încă o dată necesitatea studierii acestui subiect, aducem aminte câteva dintre cele mai importante aplicații ale cheilor:

- semnături digitale
- verificarea de semnături digitale
- autentificare
- criptarea / decriptarea datelor
- transportul cheilor
- agrearea de chei

Criptografia asigură la nivelul generării de chei, și mai departe în derivarea și schimbul acestora, faptul că informația nu a fost modificată după ce a fost trimisă, autentificând deținătorul informației, conservându-se astfel principiile de integritate, non-repudiere etc.

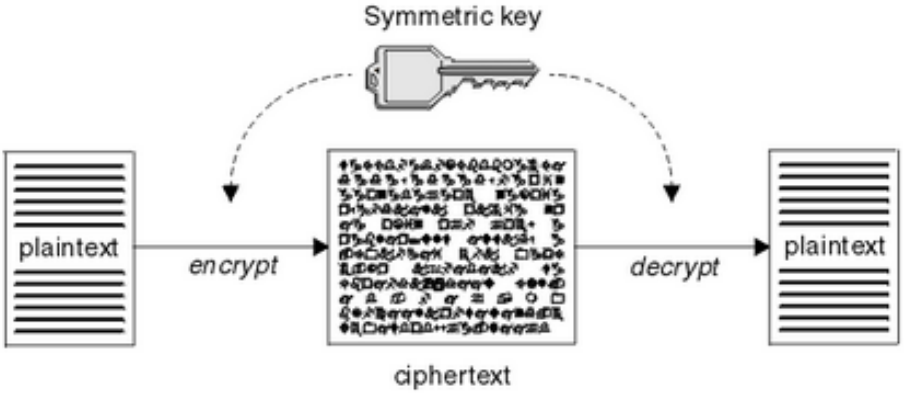
2.1 Generarea cheilor simetrice

Cheile simetrice sunt chei private $Priv_K$ ce trebuie să fie generate de una sau mai multe entități care o vor partaja pe aceasta, sau de o autoritate centrală, cheia privata $Priv_K$ fiind ulterior utilizată pentru: criptare sau decriptare (AES), generare de MAC-uri (AES) [message authentication code], derivare de chei adiționale, etc.

$$Priv_K - - - - - - - - - - > entitate_1, entitate_2, ..., entitate_n$$

Entitățile pot fi autorizate să aducă modificări asupra cheii. Mai jos vom lista o serie de utilizări sau operații posibile cu cheile simetrice, operații ce pot fi făcute folosind: funcții sau generatori random, pseudo-random, funcții de parole sau PIN, standarde de generare, generare de chei din alte chei, key agreement etc.

- generarea directă a cheilor simetrice
- distribuirea cheii simetrice generate
- generare folosind scheme de key-management
- chei simetrice derivate din chei partajate
- chei simetrice derivate din parole
- chei simetrice rezultate din combinarea mai multor chei cu alte date sau informații
- înlocuirea cheilor simetrice



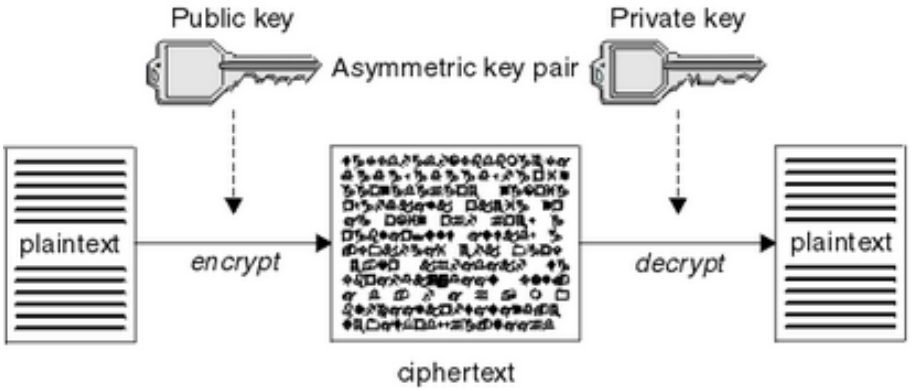
2.2 Generarea cheilor asimetrice

Cheile asimetrice sunt perechi de chei (cheie publică, cheie privată) (Pub_K, $Priv_K$) ce sunt deținute de o singură entitate, denumită și proprietarul perechii de chei, și sunt generate fie de entitate fie de către autoritatea centrală.

(Pub_K, $Priv_K$)——————————————————entitate

Cu acest prilej, ținem să listăm și aici câteva utilizări sau operații posibile cu cheile asimetrice:

- pereche de chei pentru semnături digitale: oferă autenticitatea originii, intergitatea datelor și non-repudierea (DSA, RSA, ECDSA).
- pereche de chei pentru stabilirea cheii: stabilirea unei chei presupune atât key agreement (funcție de informații ale tuturor participanților astfel încât predeterminarea cheii să fie imposibilă tuturor) cât și key transport (o entitate alege valoarea pentru cheia secretă $Secret_K$ după care o distribuie în mod sigur unei alte entități) - RSA
- distribuirea perechii de chei: cheia privată $Priv_K$ trebuie ținută secretă, cheia publică Pub_K poate fi făcută publică însă într-o manieră ce asigură intergitatea.



După această scurtă introducere, prezentăm cititorului urmatoarele subiecte de interes în acest capitol. Vom aborda generatorii și funcțiile pseudo-random, criptosistemele stream și criptosistemele bloc după care vom explica exemple de generare de chei simetrice și asimetrice.

2.3 Generatori pseudo-random

Se numește PRG (pseudo-random generator) un algoritm determinist de complexitate timp polinomială, care, pornind de la un *seed* , o samânță, de dimensiune k, s ϵ $0,1^k$, generează șiruri w_i, i ϵ [1,l(k)].

Generatorul trebuie să fie construit de așa natură încat unui posibil adversar A să îi fie indistingibil dacă w_i ϵ $0,1^k$ a fost generat de un generator PRG sau dacă a fost ales random uniform din mulțimea $\{0,1\}^l k$.
$0,1^k$ $------------>$ G(s) $-------->$ w
$0,1^l(k)$ $---------------------->$w
A(w)= ? $-------->$ $0,1^k$ $---------->$ PRG
A(w)= ? $-------->$ $0,1^l(k)$ $--->$ random uniform

Definition 2.3.1. Fie n $\geq$ 1 un numar natural (ϵ $\mathbb{N}$) si l un polinom (ϵ $\mathbb{P}$). Numim generator pseudo-random (PRG), un algoritm determinist de complexitate timp polinomială (DPT) cu factorul de expansiune l dacă sunt îndeplinite următoarele:

1. Valoarea polinomului l în n este mai mare sau egală cu numărul n, pentru orice n număr natural. l(n) $\geq$ n, $\forall$ n ϵ $\mathbb{N}$

2. Pentru orice algoritm de complexitate timp polinomială PPT D, are loc și este neglijabilă urmatoarea probabilitate: (Probabilitatea ca r să fie extras random uniform din mulțimea $\{0,1\}^l(n)$ minus probabilitatea ca s să fie extras random uniform din mulțimea $\{0,1\}^n$ este neglijabilă.

   P(D)=1: r$\leftarrow$ $\{0,1\}^l(n)$)- P(D(G(s))=1: s$\leftarrow$ $\{0,1\}^n$

- D(r)=1, înțelegem că D este un algoritm de test care decide dacă r este ales random uniform din $0,1^l(n)$.
- D(G(s)=1, întelegem că D este un algoritm de test care decide dacă G(s) este ales random uniform din $0,1^l(n)$, s ϵ $0,1^n$.

2.4 Functii pseudo-random

În această secțiune propunem să facem cunoscut cititorului urmatoarele trei teoreme, ce fac legătura între funcțiile pseudo-random și rezistența la falsificare relativ la diverse metode de verificare a autenticității mesajelor (MAC, CBC-MAC si XCBC-MAC).

Teorema 6. *F este o funcție pseudo-random (PRF), atunci înseamnă că schema MAC (Message Authentication Code) este rezistentă la falsificare.*

Teorema 7. *Dacă F este o funcție pseudo-random (PRF), atunci CBC-MAC este rezistentă la falsificări.*

Teorema 8. *Dacă F este funcție pseudo-random (PRF), atunci X CBC-MAC este rezistentă la falsificări.*

2.5 RC4 - Generator Pseudo Random

În acest subcapitol vom explica pe larg generatorul pseudo-random RC4, prezentând schema MAC, schema CBC-MAC și ulterior o posibilă implementare și un exemplu, proprietăți ale RC4 alături de avantajele și dezavantajele folosirii acestui generator.

În cele din urmă, vom aduce la cunoștință un atac important asupra acestui generator, atac ce poartă numele NOMORE ATTACK.

Schema MAC:

1. Fie F=(F_k) $\rightarrow$ o funcție PRF (pseudo-random)
   Presupunem F_k : $0,1^n \rightarrow 0,1^n$
   G = generator ce alege random uniform o cheie k
   MAC(k,m) = F_k(m), unde k = cheie, m = mesaj, t = F_k(m)
   Vrt(k,m,t) = 1 $\Leftrightarrow$ t = F_k(m)
2. F=(F_k) $\rightarrow$ o funcție PRF (pseudo-random)
   S=(G, MAC, Vrf)
   G = generator ce alege random uniform o cheie k pentru F; presupunem că n este multiplu de 4
   F_k : $0,1^n \rightarrow 0,1^n$
   m=$m1, m2, .., m_l$, $|m_i|$=n/4
   l se scrie pe n/4 biti
   Tag-ul pentru m_i se obține prin : r $\leftarrow 0,1$pow$(n/4)$, $t_i = F_k$ (r || l || [i] || m_i)
   t = (r, t_1, ..., t_l) reprezintă tag-ul pentru m.
   Dezavataje: tag scurt, pentru mesaje de lungime $\nmid$ (n/4) $\rightarrow$ padare.

Schema CBC-MAC:

1. Rezolvă problema tag-urilor pentru mesaje de lungime variabilă fără padare.
   Presupunem
   m = ..m_1.....m_2 ..... m_l
   0..0 $\rightarrow \oplus \rightarrow \oplus \rightarrow$...$\rightarrow \oplus$
   ..... $\downarrow$ ..... $\downarrow$ ..... $\downarrow$
   ..... t_1 ... t_2 ..... t_l
   MAC(k,m)=t_l

2. X CBC-MAC:
   F=(F_k) $\rightarrow$ o funcție PRF (pseudo-random)
   3 chei: K_1, K_2 si K_3; ultimele două fiind pentru padare
   Cazul 1: $|m_l| = |m_{l-1}| \rightarrow$ nu necesită padare. $t_l = F_k$ ($t_{l-1} \oplus m_l \oplus k_2$)
   Cazul 2: $|m_{l-1}| > |m_l| \rightarrow$ se padează cu m_l cu 10...0 $t_l = F_k$ ($t_{l-1} \oplus m_l$ 10..0 $\oplus$ k_3)

În cadrul generatorilor pseudo-random, putem aminti de RC4, criptosistem stream menționat în cadrul capitolului anterior, la secțiunea protocolului SSL&TLS, iar în cadrul funcțiilor pseudo-random amintim de AES si DES, criptosisteme bloc, menționate de asemenea în cadrul capitolului anterior la secțiunea protocolului SSL&TLS.

2.5.1 RC4:

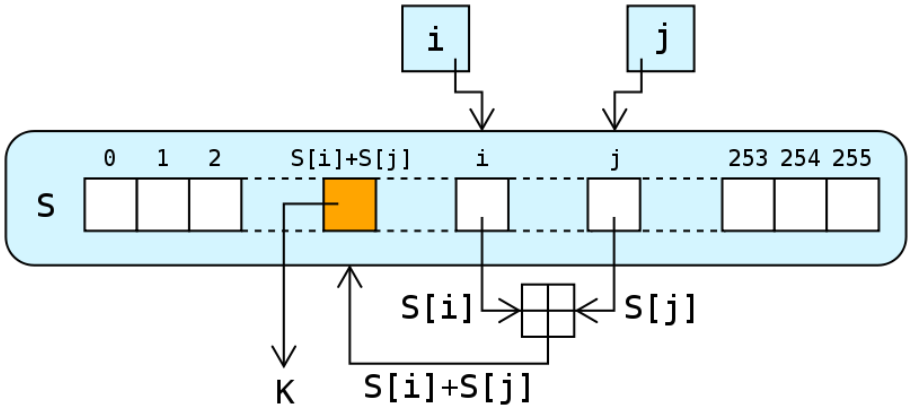
RC4 (Rivest Cipher 4), este un criptosistem stream (de chei simetrice) în care plaintext-ul este combinat cu un număr pseudo-random, astfel încat fiecare bit din plaintext împreună cu bitul corespunzător al cheii, dau bitul corespunzător criptosistemului.
RC4 mai este cunoscut și sub numele de ARC4 sau ARCFOUR.

Algoritmul :
i:=0
j:=0
while Generating Output:
——i:=(i+1) mod 256
——j:=(j+S[i]) mod 256
——swap value of S[i] and S[j]
——k:=S[S[i]+S[j]) mod 256
——output k
endwhile

CRIPTAREA :

..Plaintext.. ⊕ ..Keystream..

→ ..Ciphertext..



EXEMPLU :

→ Reset i=j=0, S=2,1,3,0
→ i=i+1=1
→ j=j+S[i]=0+1=1
→ Swap S[i] and S[j] : S=2,1,3,0
→ Output z=S(S[i]+S[j])=S[2]=3
→ z=3 (0000 0011)
→ H 0100 1000
→ 3 0000 0011 ⊕
→ ––––––
→ 0100 1011
→ i=1, j=1, S=2,1,3,0
→ i=i+1=2
→ j=i+S[i]=1+3 (mod 4) = 0
→ Swap S[i] and S[j] : S=3,1,2,0
→ Output z=S(S[i]+S[j])=S[1]=1
→ z=1 0000 0001
→ 0100 1001
→ 0000 0001 ⊕
→ ––––––
→ 0100 1000

Rezultat :
plaintext: 0100 1000 0100 1001
ciphertext:0100 1011 0100 1000

Proprietăți :
Câteva proprietăți ale algoritmului ce merită a fi menționate sunt faptul că este un generator stream-cipher de cheie variabilă ca dimensiune, este un algoritm de generare pseudo-random, key-scheduling, returnând numere pseudo-random ca

byte streams. Generatorul este testat atât pentru criptare cât și pentru decriptare.

Avantaje :
Avantajele algoritmului sunt rapiditatea față de algoritmul DES, spațiul de chei foarte mare ($\sim$ 1700 biti) și faptul că este folosit în cadrul protocolului SSL pentru protejarea traficului de Internet în WEP (Wired Equivalent Privacy) pentru securizarea rețelelor wireless. (Alte protocoale bazate pe RC4: WPA - WiFi Protected Access, BitTorrent protocol encryption, SSH - Secure Shell, RDP - Remote Desktop Protocol, Kerberos, etc)

Dezavantaje :
În ceea ce priveste dezavantajele acestui generator, putem afirma că primii 3 bytes ai output-ului dezvăluie informații despre cheie, algoritmul deține un număr mare de chei slabe (1/256), care pot fi detectate și exploatate cu o probabilitate ridicată.

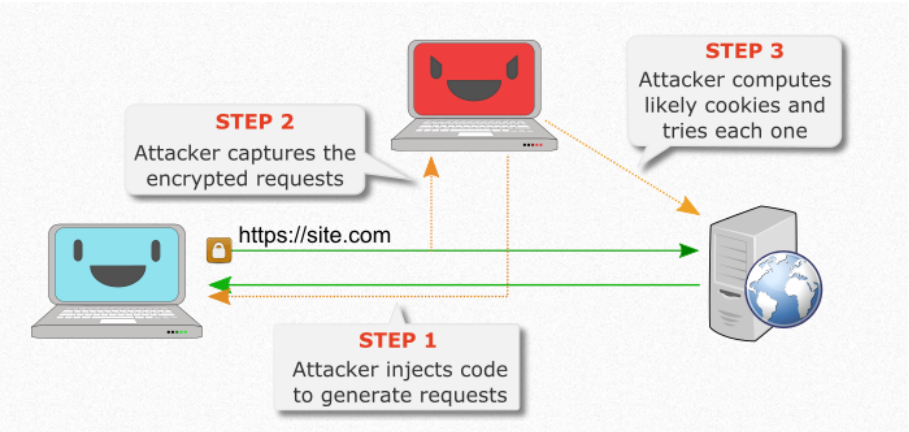
VarianteRC4 :
Diverse variante ale algoritmului au fost propuse, însa cele mai remarcate au fost RC4A (Varianta propusă de Souradyuti Paul și Bart Preneel, aduce doar o îmbunătățire a vitezei), RC4+ (key-schedule mai complex decat în RC4, mai greoi de 1.7 ori decât RC4.) și Spritz (Ron Rivest si Jacob Schuldt, algoritmul poate fi folosit atât ca generator pseudo-random cât și ca funcție hash).

Atacuri :

De-a lungul timpului au existat mai multe atacuri precum: Fluher Mantin Shamir (2001), Klein (2005), Royal Holloway (2013), Bar-mitzwah (2015) cel din urmă fiind NOMORE (2015).

În cele ce urmează, pentru a încheia prezentul capitol, vom face cunoscut cititorului atacul NOMORE.

NOMORE = Numerous Occurrence MOnitoring and Recovery Exploit. Atacul a fost făcut datorită cercetătorilor în securitatea informației din KU Leuven, Universitate din Belgia, care au prezentat noi atacuri împotriva RC4 atât asupra TLS cât și WPA-TKIP.



NOMORE este primul atac de acest gen pus în practică, asupra TLS-ului decriptând un cookie HTTPS în 75 ore iar asupra WPA-TKIP permitând atacatorului să decripteze și să injecteze pachete arbitrare în doar o oră.

O criptanaliză a RC4 arată că în TLS și WPA descoperă 220 din 256 bytes ai plaintext-ului după trimiterea aceluiași mesaj de 1.000.000.000.000 ori, iar unii bytes pot fi redobândiți abia după 16.000.000.000.000 transmisiuni.

3. Derivarea cheilor

3.1 Descriere generală a KDF(key derivation function)

O $\underline{KDF = KeyDerivationFunction}$ este o funcţie pseudo-random cu ajutorul căreia una sau mai multe chei secrete (sk), aparţinând unor parole, master keys, etc., sunt derivate.

Scopul KDF-urilor este evitarea "cheilor slabe" în sistem, lungirea cheilor la o dimensiune dorită sau adecvată sistemului, obţinerea acestor chei într-un anumit format pentru a se potrivi protocoalelor de agreare de chei.

Funcţiile de derivare a cheilor KDF sunt de obicei utilizate împreună cu parametri publici, pentru a preveni un atacator să obţină derivarea prin învaţarea unor informaţii critice legate fie de inputul valorii secrete fie legate de alte chei derivate anterior.

Introducem noţiunea de $\underline{KS = KeyStretching}$ care este o tehnică de securizare a posibilelor chei slabe (parole sau passphrase) împotriva atacurilor brute-force (passwod cracking; funcţiile sunt în mod deliberat greu de calculat (ca timp) pentru prevenirea unor atacuri precum cel menţionat anterior sau precum dictionary attack.), prin creşterea timpului de testare a tuturor posibilităţilor unei chei. Key strengthening extinde cheia cu un salt random şi, spre deosebire de key stretching, şterge ulterior în mod sigur saltul adăugat.

```
DK = KDF ( Key , Salt , Iterations )
```

1. DK = cheie derivată
2. KDF = funcţie de derivare a cheii
3. Key = cheia iniţială
4. Salt = parametru ne-secret , număr random (utilizarea acestui parametru previne precalcularea unui dicţionar de chei derivate [dictionary attack])
5. Iterations = numărul de iteraţii al unei sub-funcţii (dificultatea atacului creşte proporţional cu numărul de iteraţii)

Cea mai importantă aplicaţie a KDF-urilor este **password hashing** -> verificarea parolelor, iar printre algoritmii de derivare a cheilor având şi proprietăţi de password hashing, care s-au remarcat, enumerăm pe următorii:

1. **Argon2** = câştigătorul competiţiei "Password Hashing Competition" 2015
   Problemele existente înaintea apariţiei algoritmului Argon 2:

   - Independente de input: în primele scheme, în care locaţia memoriei citite era cunoscută în avans, a devenit vulnerabilă la atacurile time-space tradeoff - un atacator poate precalcula blocul lipsă până acesta să fie generat sau cerut de sistem.
   - Dependente de input: vulnerabil la side-channel attack.
   - Cât de mare ar trebui sa fie un bloc de memorie? Blocuri mici - încetinire din cauza principiulului de cache CPU. Blocuri mari - greu de procesat din cauza numărului limitat de regiştri lungi.
   - Dacă blocul de memorie este mare, cum să alegem funcţia de compresie?
   - Cum să exploatăm core-uri multiple de procesor?

   Soluţia Argon2: arhitectura x86 ce exploatează cache-ul şi memoria procesoarelor Intel şi AMD.

   - *Argon2d*: mai rapid, utlizând accesul memoriei depinzând de date - perfect pentru aplicaţii fără ameninţări din partea side-channel timing attack
   - *Argon2i*: mai încet, efectuând mai multe operaţii pe memorie pentru a oferi securitate împotriva tradeoff attack.

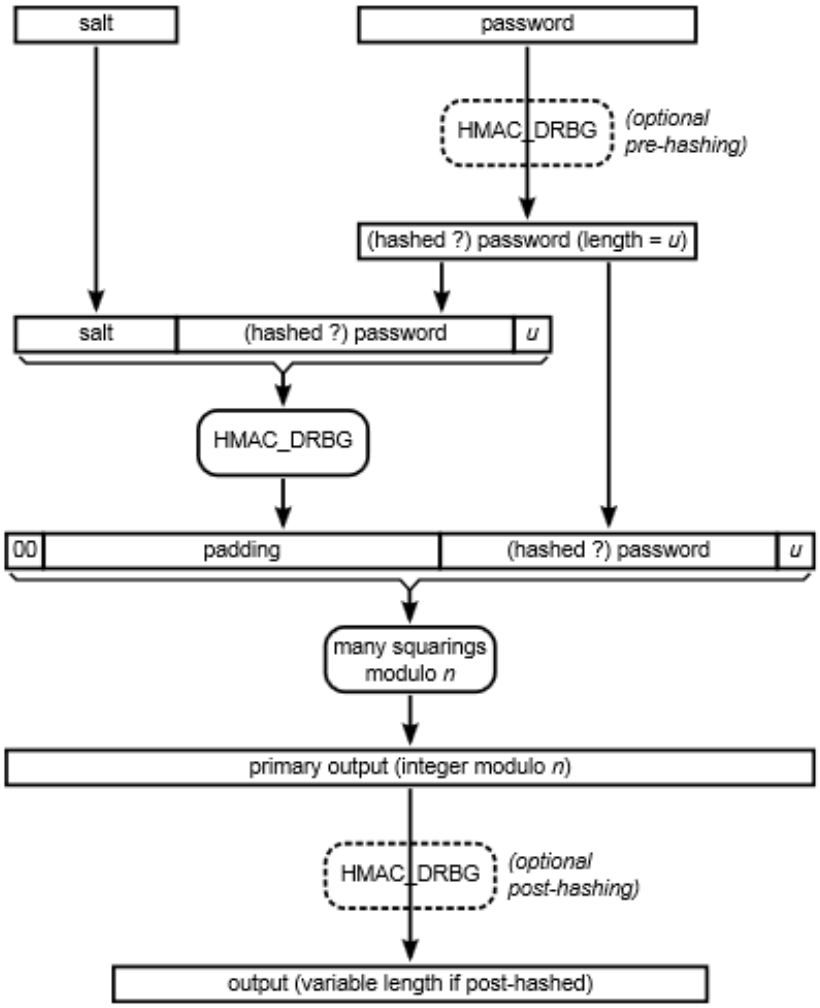
2. **Catena** = primul password scrambler care satisface atât consumul de memorie, prevăzând în acelaşi timp rezistenţa împotriva atacurilor cache-timing. Este un algoritm sigur, asigură indistingibilitate la numerele random şi este rezistent la side-channel attack.

Algorithm	Cost Factor (default)	Memory	Server Relief	Client-Indep. Updates	Issues
`crypt` [35]	25	small	-	-	"too fast"
`md5crypt` [25]	1,000	small	-	-	"too fast"
`sha512crypt` [13]	1,000–999,999 (5,000)	small	-	-	(small memory)
`NTLMv1` [20]	1	small	-	-	"too fast"
`PBKDF2` [56]	1–∞ (1,000)	small	-	-	(small memory)
`bcrypt` [44]	2^4–2^{99} (2^6, 2^8)	4,168 bytes	-	-	(constant memory)
`scrypt` [43]	1–∞ (2^{14}, 2^{20})	flexible, big	-	-	cache-timing attacks
`Catena-BRG` (this paper)	2^1–∞ (2^{17}, 2^{20})	flexible, big	✓	✓	performance
`Catena-DBG` (this paper)	2^1–∞ (2^{14}, 2^{16})	flexible, big	✓	✓	performance

   Comparaţie a scrambelelor parolelor state-of-the-art şi CATENA. Toţi algoritmii menţionaţi suportă valori de salt. Termenul 2^1 - ∞ înseamnă factorul de cost care poate fi ales arbitrar de mare.

3. **Lyra2** = abilitatea de a configura cantitatea de memorie dorită, timpul de execuţie, gradul de paralelism [numărul de thread-uri], lungimea output-ului etc.; decuplarea costului memoriei de costul timpului; rezistent la time-memory trade-offs attack şi side-channel attack.

4. **Makwa** = transformă un input de lungime variabliă într-un output de lungime fixă. Oferă încetinire configurabilă dar şi salt-ing. Diferit faţă de ceilalţi algoritmi prin faptul că hashing-ul poate fi delegat unui sistem extern nesigur. Ideea principală este că acest sistem este văzut ca un atacator pasiv, unde se vor aplica toate funcţiile necesare şi se va încerca observarea datelor de către acest atacator pasiv, scopul fiind creşterea puterii computaţionale a password hashing-ului.



Asupra parolei este aplicat opţional o funcţie hash, după care se padează determinist (folosind salt-ul şi funcţia KDF). Asupra valorii finale se aplică apoi opţional o funcţie hash.

Scenarii de delegare: Password Authenticated Key Exchange and Small Clients, Feeble Server and the Cloud, Heterogenous Clients.

5. **Yescrypt** = mass-user authentication: reţele sociale, portaluri web, bancă, cumpărături online, servicii de plată, conturi de domeniu, derivare de chei etc.

Facem cunoscut cititorului faptul că o funcţie de derivare a cheii impropriu definită, poate face materialul de derivare vulnerabil la atacuri.

Spre exemplu, securitatea globală a materialului de derivare a cheii depinde foarte mult de protocoalele ce stabilesc cheia de derivare a cheii, protocoale stabilite la nivel de funcţie KDF.

De asemenea, notăm faptul că puterea sau securitatea unei chei este măsurată de proprietatea de indistingibilitate faţă de adversar. Dat un adversar A, acestuia îi este imposibil a determina dacă output-ul a fost generat de o funcţie KDF sau de o distribuţie uniformă de string-uri de aceeaşi lungime de biţi ca şi KDF-ul, cu menţiunea că cheia de derivare a cheii este singura intrare necunoscută.

Dat aşadar un set de date de intrare, diferit de cheia de derivare a cheii, aceasta din urmă poate fi aflată de adversar în 2^w execuţii ale KDF-ului, unde w este lungimea în biţi a cheii de derivare a cheii.

3.2 PBKDF

O aplicaţie ce derivează chei în modul Password-based KDF, trebuie să urmărească urmatorii paşi:

1. Selectează un salt S şi un counter de iteraţie c.
2. Selectează lungimea în octeţi a cheii de derivare.
3. Aplică KDF (parola, salt, counter, lungime)=DK.
4. Afisează cheia derivată.

```
DK = KDF ( PRF, Password , Salt , Iterations , dkLen )
```

- DK = cheia derivată
- KDF = funcţia de derivare a cheii
- PRF = funcţie pseudo-random
- Password = parola din care se derivează o cheie
- Salt = secvenţa random de biţi
- Iterations = numărul dorit de iteraţii
- dkLen = lungimea dorită a cheii derivate

DK = $T1 \parallel T2 \parallel ... \parallel TdkLen/hLen$
Ti = F (Password , Salt , Iterations , i)
F = XOR
Exemplu de derivare de cheie în protocolul WPA2 - Wifi-Protected Access:

```
DK = PBKDF ( HMACSHA1 , passphrase , ssid , 4096 , 256)
```

PBKDF poate fi implementat cu un circuit mic şi puţin RAM, ceea ce face ca un atac brute-force să fie ieftin, folosind circuite integrate sau unităţi de procesrare grafică.
În urma competiţiei amintitite şi în secţiunea 3.1, "Password Hashing Competition", Argon2 a fost ales pentru derivare.

Diferenţele dintre PBKDF1 si PBKDF2 se observă la nivelul funcţiilor aplicate. În timp ce prima aplică o funcţie hash, cea dea doua foloseşte o funcţie pseudorandom. De asemenea, PBKDF1 produce chei ce nu sunt mereu suficiente ca lungime tuturor aplicaţiilor, în timp ce PBKDF2 nemărgineşte lungimea cheilor, însa le limitează la structura funcţiei pseudorandom.

3.3 HKDF

HKDF = HMAC-based Extract-and-Expand KDF

Transformă orice date slabe în material criptografic sigur (exemplu: convertirea secretelor pratajate Diffie-Hellman în material potrivit pentru criptare, verificare a integrităţii sau pentru autentificare).

HMAC-based Extract and Expand KDF este deja utilizat în diverse protocoale precum IKEv2 (Internet Key Exchange), PANA (Protocol for Carrying Authentication for Network Access) şi EAP-AKA (Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement).

HKDF urmează paradigma "extrage apoi extinde", în care funcţia de derivare a cheii este compusă din două module:

1. Primul modul preia input-ul de material al cheii şi extrage din acesta o cheie K pseudo-random de lungime fixă. (IKM = input keying material ; PRK = pseudorandom key)

   ```
   HKDF-Extract(salt, IKM) → PRK
   ```

2. Al doilea modul extinde cheia K în mai multe chei pseudo-random adiţionale, având dimensiunea dorită. (info = informaţii specifice aplicaţiei - opţional ; L = lungimea output-ului ; OKM = output keying material)

   ```
   HKDF-Expand(PRK, info, L) → OKM
   ```

Aplicaţii ale HKDF-ului:

- Construirea de generatori pseudo-random pentru surse imperfecte de randomness (RNG fizic)
- Construirea de generatori pseudo-random din surse slabe de randomness (entropii colectate din evenimente ale sistemului)
- Derivarea de chei criptografice dintr-o valoare partajată Diffie-Hellman într-un protocol de key-agreement
- Derivarea de chei simetrice dintr-o schemă hibridă de criptare publică.

Teorema 9. *Fie H funcţia hash Mercke-Damgard, construită pe o familie pseudo-random de funcţii de compresie { hK } K. Fie S o colecţie de distribuţii de probabilitate comportându-se ca o sursă de material de chei. Presupunând că instanţierea HMAC-ului cu familia de funcţii de compresie prezentată mai sus este un extractor computaţional sigur al sursei S, putem afirma că HKDF este o funcţie de derivare a cheilor sigură în relaţie cu S.*

O funcţie KDFHMAC este alcatuită, ca şi după acronim, din două parţi:

- MAC (Message Authentication Code) [generare ; verificare]
- KDF (Funcţie de derivare a cheilor)

3.3.1 Generarea MAC-urilor

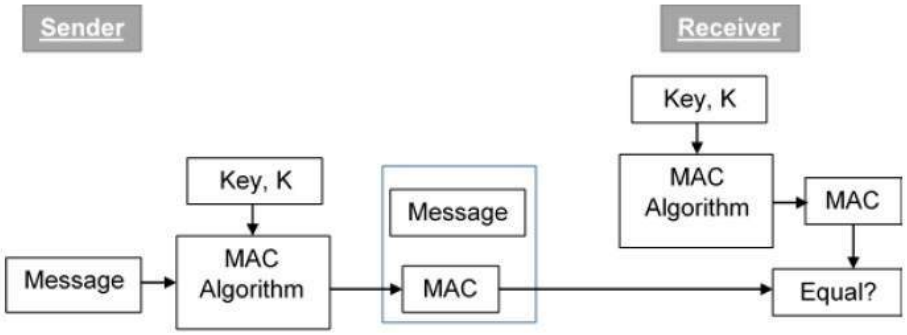
Generarea MAC-urilor produce un cod de autentificare a mesajelor dintr-un mesaj cu cheia K. Paşii necesari generării unui cod de autentificare a mesajelor sunt următorii:

1. Se stabileşte S = saltul, c = counterul
2. Se stabileşte lungimea cheii n octeţi
3. Se aplică funcţia de derivare a cheii KDF(parolă, salt, counter)D=DK
4. Se procesează mesajul m cu schema MAC cu cheia DK pentru a genera un cod t de autentificare a mesajului.
5. Afisează codul t de autentificare a mesajului

3.3.2 Verificarea MAC-urilor

Verificarea MAC-urilor presupune certificarea codului t generat anterior asupra mesajului m cu cheia DK. Paşii pentru certificarea codului de autentificare a mesajelor sunt următorii:

1. Obţinem saltul S şi counterul c
2. Obţinem lungimea cheii
3. Aplicăm KDF(parolă, salt, counter) pentru a produce un DK de lungimea dorită şi necesară
4. Procesează mesajul m pentru a verifica t
5. Daca codul t este corect, → "correct", altfel, "incorrect"



3.4 ConcatKDFHash - ConcatKDFHMAC

INPUT :

- Z : byte string ce reprezintă un secret partajat
- keydatalen : integer ce defineşte lungimea în biţi a materialului secret de cheie ce trebuie să fie generat (ar trebui să fie $\leq$ hashlen * $(2^{32} - 1)$)
- OtherInfo : bit string egal cu concatenările următoare:
  $AlgorithmID \| PartyUInfo \| PartyVInfo \| SuppPubInfo \| SuppPrivInfo$
  1. AlgorithmID = bit string ce indică cum va fi parsat materialul derivat de cheie şi pentru ce algoritmi va fi folosit acest material.
  2. PartyUInfo = bit string conţinând informaţii publice cerute de aplicaţie folosind KDF ce contribuie la un terţ U la procesul de derivare a cheii.
  3. PartyVInfo = bit string conţinând informaţii publice cerute de aplicaţie folosind KDF ce contribuie la un terţ V în procesul de derivare a cheii
  4. SuppPubInfo = bit string conţinând adiţional informaţii publice cunoscute mutual.
  5. SuppPrivInfo = bit string conţinand adiţional informaţii private cunoscute mutual (exemplu: cheie secretă simetrică partajată comunicată într-un canal separat)

OUTPUT :
DerivedKeyingMaterial = bit string de lungime keydatalen (sau un indicator de eroare în cazul eşecului de derivare).

PROCES :

- reps = [keydatalen / hashlen]
- Dacă reps $\geq (2^{32} - 1)$), atunci abortează, afisează un indicator de eroare şi opreşte-te
- Iniţializează un byte string de 32 biţi în big-endian ca 00000001 în baza 16.
- Dacă counter || Z || OtherInfo este $\geq$ max_hash_inputlen, atunci abortează, afișează un indicator de eroare şi oprește-te
- pentru i=1:
  - calculează Hi = H (counter || Z || OtherInfo)
  - creşte counter (modulo 2^{32}) tratându-l ca un integer pe 32 biţi neasignat.
  - fie Hhas un set de Hash reps dacă (keydatalen / hashlen) este un integer; altfel, fie Hhash un set de (keydatalen mod hashlen) cei mai din stânga biţi ai Hash reps.
  - setează DerivedKeyingMaterial = $H1$ || $H2$ || ... || $Hashreps - 1$ || Hhash

Nici ConcatKDFHash nici ConcatKDFHMAC nu ar trebui folosite pentru stocarea parolelor.

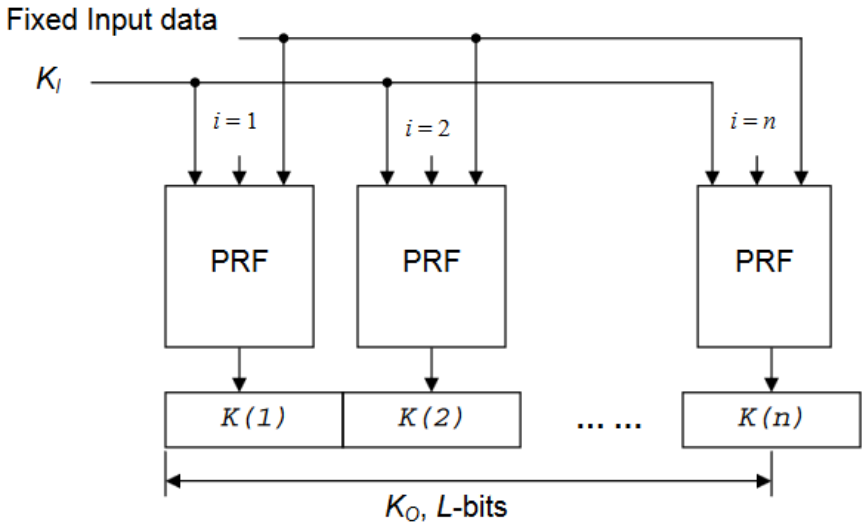
3.5 Moduri ale KDF

Aşa cum am văzut în subsecţiunile anterioare, o funcţie de derivare a cheii iterează o funcţie pseudorandom de un anumit număr de ori, concatenând output-ul până când se ajunge la lungimea dorită de material de cheie.

În funcţie de modul de iterare peste funcţia pseudorandom, identificăm trei tipuri de funcţii KDF:

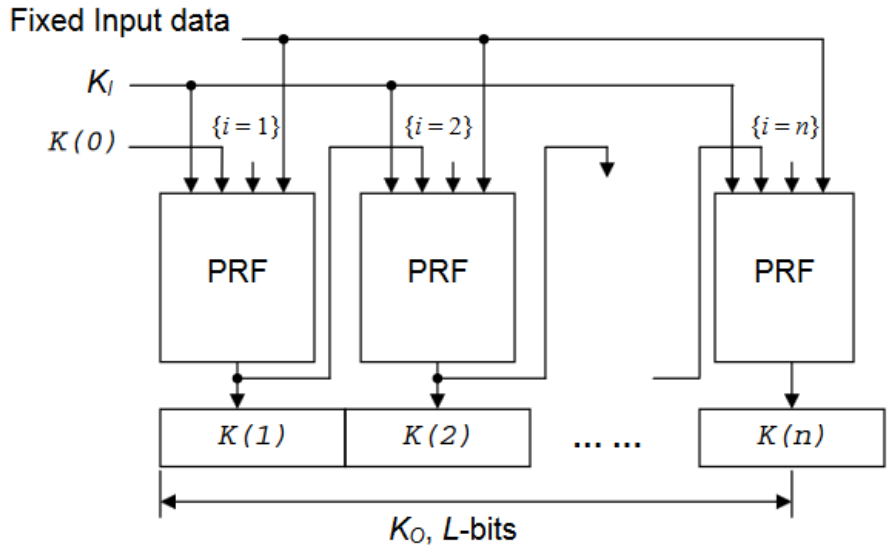
3.5.1 Counter Mode KDF

În modul counter, funcţia pseudorandom este calculată cu un counter ca variabilă de iteraţie (Funcţia pseudorandom este iterată în single pipeline).



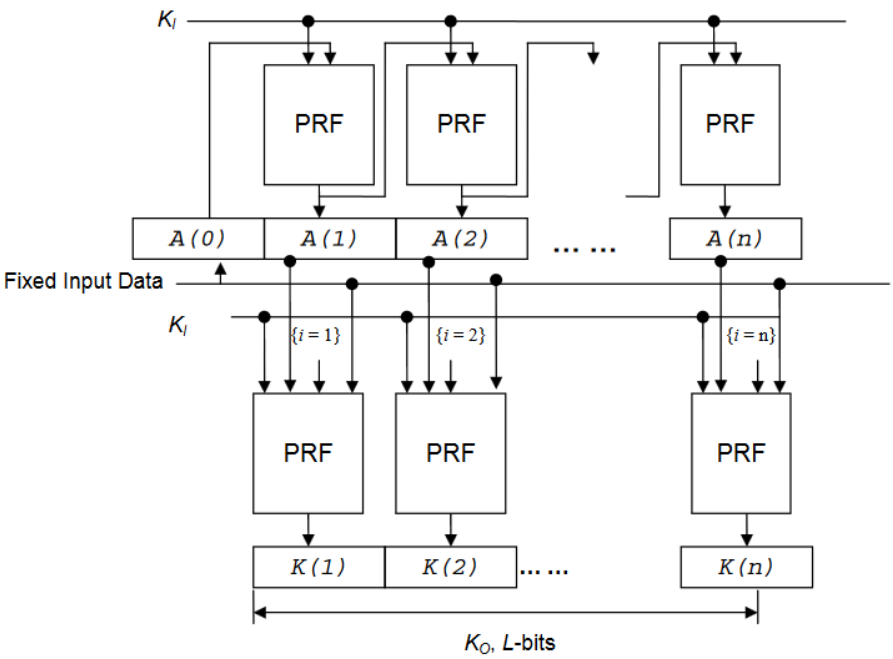
3.5.2 Feedback Mode KDF

În modul feedback, output-ul funcţiei pseudo-random este calculat folosind rezultatul iteraţiei precedente şi opţional, un counter ca variabilă de iteraţie. (Funcţia pseudorandom este iterată în single pipeline).



3.5.3 Double Pipeline Iteration Mode KDF

Spre deosebire de modelele anterioare, funcţia pseudo-random iterează în două pipeline-uri. Prima iteraţie generează o secvenţa de secrete folosite ca input pentru al doilea pipe-line la acelaşi număr al iteraţiei.


K_I
PRF
PRF
PRF
A(0)
A(1)
A(2)
... ...
A(n)
Fixed Input Data
K_I
{i = 1}
{i = 2}
{i = n}
PRF
PRF
PRF
K(1)
K(2)
... ...
K(n)
K_O, L-bits

4. PUF - Physically Unclonable Functions

Primul capitol al lucrării de față aduce la cunoştinţă cititorului faptul că domeniul criptografiei şi al securității informatice prezintă un mare interes, având aplicabilitate în multe domenii.

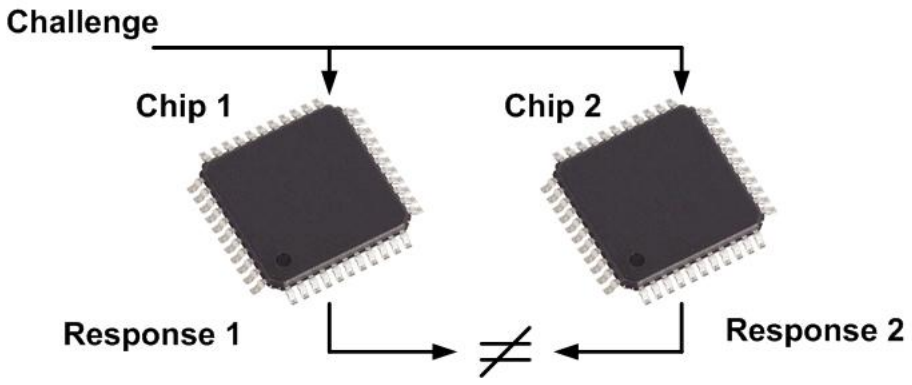
Capitolul prezent va face legătura cu domeniul Smart Card-urilor, relativ la PUF-uri, ce promit a asigura o "amprentă" prin generarea de chei criptografice unice fiecărui smart card în parte.

O functie PUF trebuie să deţină urmatoarele proprietăţi:

- *Evaluabilă*: uşor de evaluat.
- *Unică*: conţine informaţii legate de device-ul asociat.
- *Nereproductibilă*: reproductibilă cu eroare foarte mică.
- *Neclonabilă*: greu de construit o procedură identică.
- *Nepredictibilă*:greu de prezis cu o eroare foarte mică.
- $One - way$
- *Falsificări*: alterând device-ul, funcţia se transformă cu o probabilitate mare.

Având această introducere, putem trage concluzia că nevoia PUF-urilor este argumentată de stocarea informaţiilor digitale pe un device într-o manieră rezistentă la atacuri fizice, indiferent de dificultatea şi costurile impuse.

Chiar dacă există două cip-uri identice, vom explica mai jos modalitatea prin care acestea nu pot fi clonate.



4.1 Tipurile principale de PUF-uri

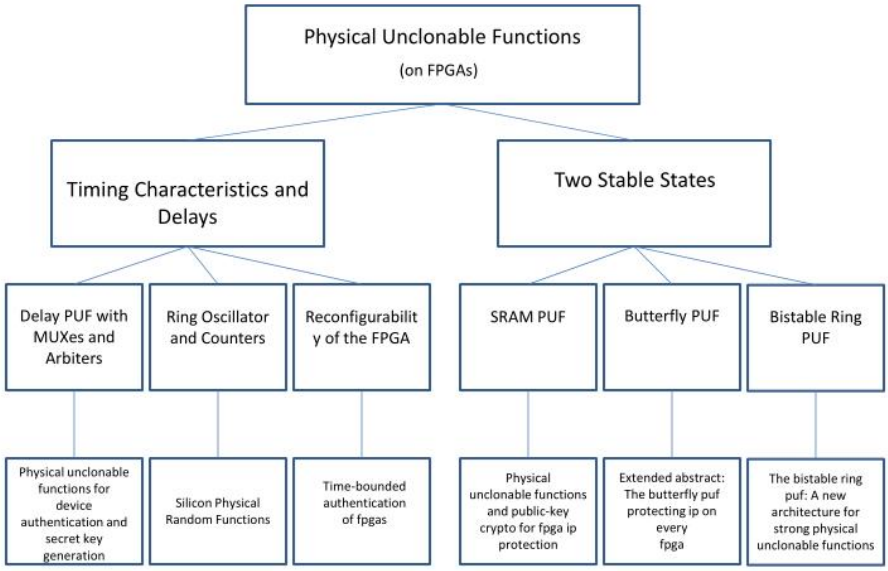
PUF-urile pot fi clasificate în două mari categorii, în funcţie de randomitatea oferită:

1. PUF-uri ce folosesc elemente random introduse explicit
    - PUF-uri optice
    - PUF-uri în straturi
2. PUF-uri ce folosesc randomitate intrinsecă
    - PUF-uri decalate
    - PUF-uri SRAM (Static Random Access Memory)
    - PUF-uri fluture

De asemenea, o clasificare a funcţiilor fizice neclonabile este descrisă în imaginea de mai jos. În funcţie de caracteristici ale timpului şi decalaje ale timpului, se identifică trei tipuri de PUF-uri: decalate cu arbitrii şi MUX-uri, inele oscilatoare şi countere, FPGA reconfigurabile [Field-programmable gate array care este un circuit integrat]). Acestea asigură atât autentificarea pe device-uri cât şi generarea de chei secrete.

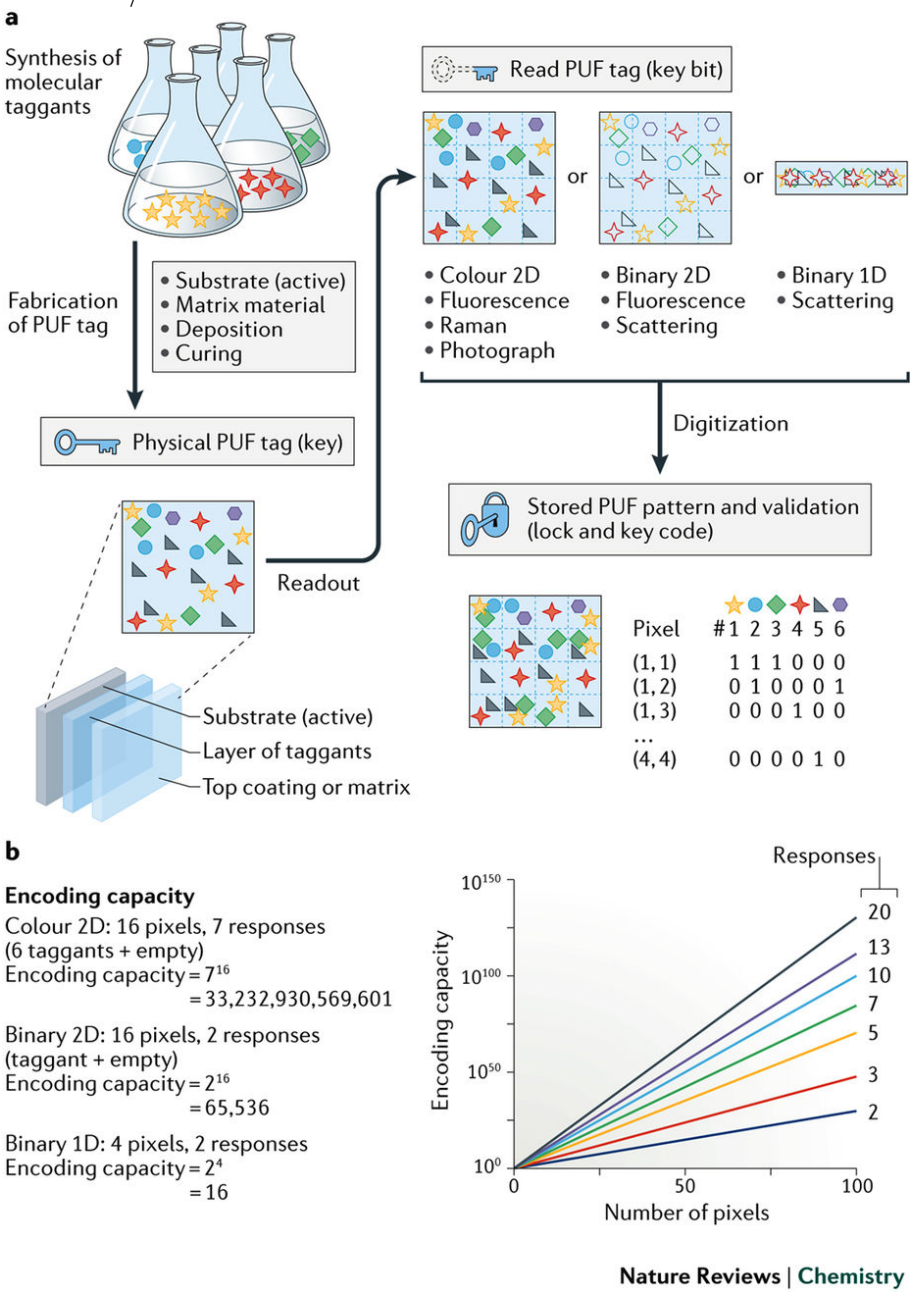
În funcţie de două stadii stabile ale PUF-urilor se identifică PUF-uri SRAM, PUF-uri fluture şi PUF-uri inel bistabile. Aplicaţii ale acestora sunt protejarea IP-ului peste un FPGA cât şi dezvoltarea unei noi arhitecturi pentru PUF-urile "strong".

În cele ce urmează, vom descrie câteva dintre cele mai importante tipuri de funcţii fizice neclonabile.



4.1.1 PUF-uri optice

PUF-urile optice sunt sisteme bazate pe fenomene optice ce deţin un mecanism care stabileşte cu exactitate orice fenomen optic.
Se stabileşte poziţia relativă a token-ului de luminare a laserului şi a camerei asupra fiecărui I/O.



4.1.2 PUF-uri decalate

PUF-urile decalate exploatează variaţiile random ale firelor, cablurilor şi porţilor din silicon. Având un challenge la intrare, circuitul porneşte având două thread-uri sau tranziţii ce propagă challenge-ul pe două căi diferite pentru a vedea pe care cale ajunge primul la final. Un "arbitru" analizează primul ciruit terminat cu 1 şi ultimul cu 0.

În momentul în care un circuit similar este fabricat pe alt cip, funcţia logică implementată de circuit este diferită din cauza sau datorită decalajelor de variaţie.

4.1.3 PUF-uri SRAM (Static Random Access Memory)

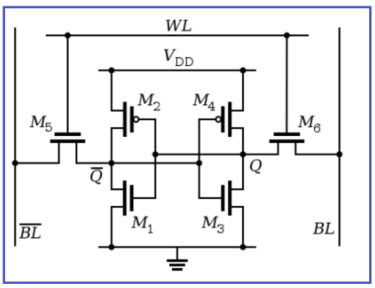
PUF-urile SRAM se află într-o cercetare continuă exploatându-se posibilele aplicaţii pentru prevenirea contrafacerii produselor sau a storage-ului sigur de chei criptografice.

Un exemplu în acest sens este tag-ul RFID (Radio-frequence Identification) [câmpuri electromagnetice pentru identificarea automată şi tagarea produselor] ce poate fi foarte uşor clonat în absenţa unui PUF. În prezenţa acestuia, clonarea într-un timp rezonabil este extrem de dificilă.

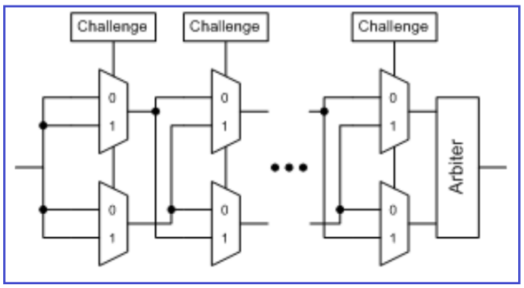
4.1.4 PUF-uri fluture

Ideea de bază a PUF-urilor tip fluture este strâns legată de cea a PUF-urilor SRAM descrise anterior. Acestea sunt compuse din două componente integrate: un şir de celule PUF (formate din lacăte cuplate încrucişate) şi un procesor.

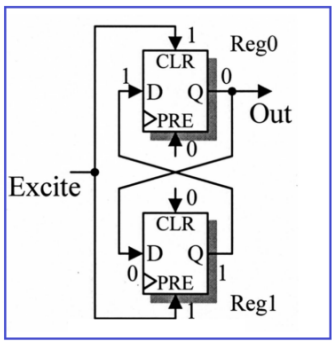
Celula este provocată prin aducerea sistemului într-o stare instabilă, lăsând-o să conveargă într-un interval de timp specific la o stare stabilă. Starea stabilă preferenţială este determinată de diferenţa definită de variaţiile procesului în timpul fabricaţiei.



SRAM PUF



DELAY: ARBITER PUF



BUTTEFLY PUF

4.2 Aplicaţii ale PUF-urilor

Până în prezent, s-au remarcat deja foarte multe aplicaţii ale funcţiilor fizice neclonablie (PUF):

1. Extragerea de chei criptografice
    - Foloseşte la protejarea împotriva execuţiei de cod nedorit.
    - O cheie criptografică derivată de un PUF al unui semiconductor, poate fi folosită la decriptarea datelor de configurare.
2. Autentificarea obiectelor
    - Diverse tehnologii sunt deja folosite pentru autentificarea produselor, precum aplicarea unui marker vizibil sau invizibil folosind etichete de securitate (cerneală / holograme) ce integrează funcţionalitatea de autentificare.
    - Un obiect sau circuit digital poate fi aşadar identificat printr-un SN (serial number), având în spate un sistem de autentificare bazat pe chei secrete sau private.
    - În spate este realizat un challenge de tipul întrebare răspuns, prin care obiectul se autentifică parţii ce îl verifică şi invers.
3. Marcarea produselor originale
    - În China, Camerele foto Canon 60D DSLR folosesc un mecanism de marcare a produselor originale pentru a le proteja împotriva contrafacerii.

Pe lângă aceste aplicaţii ale PUF-urilor putem enumera: storage sigur pentru chei secrete, identificator al sistemelor hardware care au integrat un circuit PUF, protocoale de transfer complexe etc.

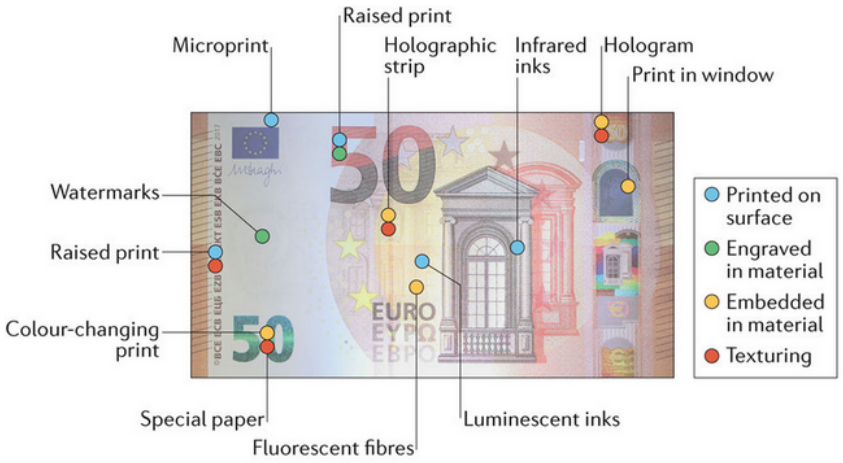
4.2.1 Abordări anti-fraudă

Un exemplu de fraudă foarte cunoscut în lume este falsificarea bancnotelor, în ciuda tuturor elementelor de securitate pe care acestea le deţin deja.

Legătura cu funcţiile fizice neclonabile descrise anterior în acest capitol este faptul că, în ciuda benzilor magnetice holografice sau a numărului de serie, fabricarea bancnotelor este supusă unui proces reproductibil, deci, pot fi contrafăcute.

Figure 1: Clonable anti-counterfeiting tags used in a euro banknote.

From: Physical unclonable functions generated through chemical methods for anti-counterfeiting



Nature Reviews | Chemistry

Soluţia ar fi aşadar introducerea unui tag anti-fraudă ce ar face o bancnotă imposibilă la clonare, precum funcţiile fizice neclonabile.

Diverse procese chimice pentru generarea unor astfel de tag-uri PUF au fost descoperite: cerneală specială, imprimări 3D biodegradabile integrate în bancnote, materiale fluorescente, tehnici de imagistică moleculară până la nanomateriale luminisciente, particule cu identitate, gravare cu laser sau control nanofotonic.

Domeniul şi studiul rămâne încă deschis, noi propuneri fiind aduse recent de către cercetători în securitate.

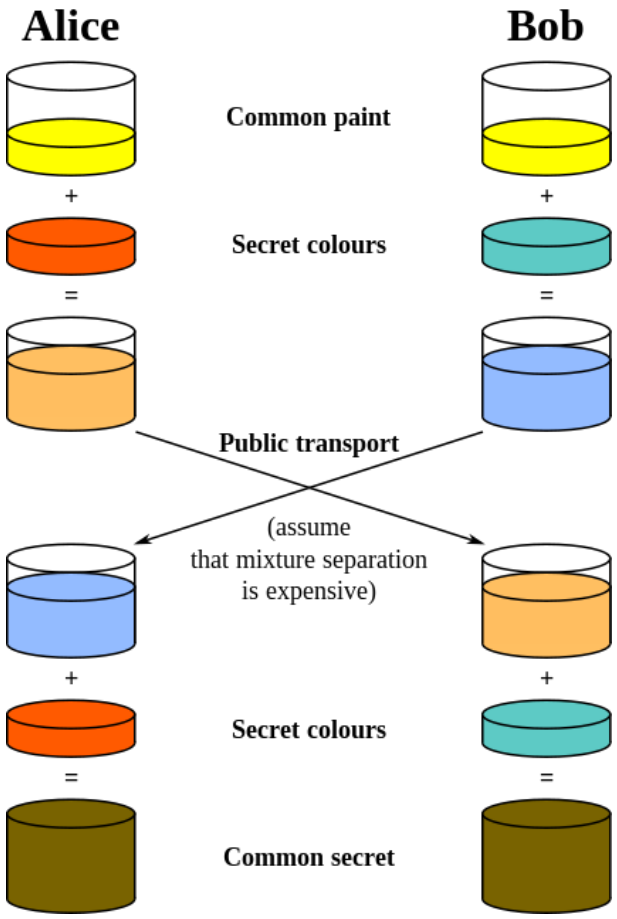
Contribuții

Contribuţiile în prezenta Lucrare de licenţă o reprezintă implementarea algoritmului Silver–Pohlig–Hellman, algoritm pentru calcularea logaritmului discret al unor numere prime foarte mari.

Legătura cu "Generarea, derivarea, schimbul şi managementul cheilor criptografice" intervine în momentul în care aplicaţia poate reprezenta un atac asupra algoritmului Diffie-Hellman, metodă de schimbare a cheilor criptografice.

Diffie-Hellman este o metodă de schimb de chei, ce stabileşte o cheie secretă partajată între două entităţi care pot fi folosite pentru o comunicare pe un canal sigur într-o reţea publică.

O astfel de schema de partajare o putem vedea descrisă în imaginea urmatoare:



Paşii algoritmului Diffie-Hellman:

- Alice si Bob agrează folosirea modulului p în baza g (rădăcina primitivă modulo p)
- Alice alege un secret a şi trimite lui Bob A = g^a mod p
- Bob alege un secret b şi trimite lui Alice B = g^b mod p
- Alice calculează s = B^a mod p
- Bob calculează s = A^b mod p
- Alice si Bob au partajat secretul s

David Wong, consultant de securitate la serviciile criptografice NCC Group, explică în lucrearea sa "How to Backdoor Diffie-Hellman" din Iunie 2016, cum poate fi atacat algoritmul Diffie-Hellman folosind tehnici de calculare a logaritmului discret.

Contribuţia presupune implementarea unui modul de calculare a logaritmului discret, mai precis a algoritmului Pohlig-Hellman în limbajul de programare Java.

Pentru a putea ataca schimbul de chei in Diffie-Hellman, un atacator ar putea extrage cheia secretă de la o cheie publică $pk1{=}g^a$ (mod p) şi apoi calcula cheia partajată g^{ab} (mod p) folosind cheia publică $pk2{=}g^b$ (mod p).

O soluţie pentru a calcula aceasta ar fi metoda "trial multiplication" însa complexitatea se ridică la q/2 operaţii pentru a găsi soluţia.

O implementare mai rapidă este abordarea algoritmilor de calculare a logaritmului discret precum "Shank baby-steps", "Shank giant-step", "Pollard rho", "Pollard Kangaroo" sau "Pohlig-Hellman", reducând astfel complexitatea la doar $\sqrt{q}$ operaţii pentru găsirea soluţiei.

Cunoscând factorizarea completă a ordinului grupului, factorii fiind relativ mici, logaritmul discret poate fi calculat cu uşurinta. Ideea principală este găsirea valorii cheii secrete modulo divizorii ordinului grupului reducând cheia publică în subgrupuri de ordine ce divid ordinul grupului.
Prin TCR (Teorema Chineza a Resturilor), cheia secretă poate fi reasamblată în ordinul grupului. Calcularea cheii secrete modulo fiecare factor *pi* la puterea *ki* al ordinului.
O modalitate de a face aceste calcule este reducerea cheii secrete la subgrup. Calculând logaritmul discret al valorii y la puterea (ϕp/(*pi* la puterea *ki*)), obţinem cheia secretă (mod *pi* la puterea *ki*).

Valoarea obţinută este un generator al subgrupului de ordin *pi* la puterea *ki* ridicat la puterea cheia secretă 1. Continuând şi aplicând teorema chineză a resturilor, obţinem cheia secretă finală.

Concluzii

Lucrarea de licenţă a adus o paralelă între Criptografie şi Securitatea Informaţiei, făcând referinţă la alte domenii precum Reţelele de Calculatoare şi Smart Cards.

Implementarea modulului Pohlig-Hellman poate servi la finalizarea atacului asupra algoritmului Diffie-Hellman, în timp ce ultimul capitol având drept subiect principal PUF-uri (Physically Unclonable Functions) reprezintă o cercetare profundă asupra unei nişe de interes ridicat.

Potenţiale direcţii viitoare de cercetare legată de tema actuală sunt dezvoltarea unui modul de integrare al celor mai importante atacuri în criptografie cât şi cercetarea unor noi metode anti-fraudă deoarece acestea înglobează nu numai Informatica dar şi Fizică şi Chimie, ştiinţe de interes personal (cu precădere fizica cuantică).

Bibliografie

[1] Katholieke Universiteit Leuven, (KUL). *Encrypt II Yearly Report on Algorithms and Keysizes (2011-2012)*. ICT-2007-216676. European Network of Excellence in Cryptography II. 5.6.7. General Key size Recommendations. p.13-34 30.09.2012.

[2] Katholieke Universiteit Leuven, (KUL). *Encrypt II Yearly Report on Algorithms and Keysizes (2011-2012)*. ICT-2007-216676. European Network of Excellence in Cryptography II. 8. Stream Ciphers. RC4 p.46-47 30.09.2012.

[3] Storage Networking Industry Association, (SNIA). *An Introduction to key Management for Secure Storage*. Key Management. Many Key Usage. p.10 2008.

[4] National Institute of Standard and Technology, (NIST). *ITL Bulletin for December 2012, Generating secure crypographic keys: a critical component of cryptographic key management and the protection of sensitive information*. Cryptographic algorythms and keys. Key generation techniques. Shirley Radack 2012.

[5] Cryptographic Extraction and Key Derivation, (IBM T.J. Watson Research Center). *The HKDF Scheme*. 4. Extract-and-Expand KDF and a HMAC-based Instantiation. p.9-11 7. The Key Expansion Module. p. 18-19 B. Sources of imperfect keying material. p. 24 E. Attack on the KDF Scheme. p. 28 2010. New York. Hugo Krawczyk

[6] Information Security Group, (Royal Holloway, University of London). *Cryptographic Key Management*. Stages in the Key Management. p. 26 Key generation. p. 27 Key usage. Keith Martin 2006.

[7] National Institute of Standard and Technology, (NIST). *Recommendation for Cryptographic Key Generation*. 7.4. Symmetric keys derrived from a pre-shared key. 7.5. Symmetric keys derrived from passwords 7.6. Symmetric keys produced by combining multiple keys and other data. p. 16-18. Elaine Barker. Allen Roginsky. Special Publication 800-133. December 2012

[8] University of Luxembourg, Luxembourg *Argon2: the memory-hard function for password hashing and other applications*. Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich December 26, 2015

[9] National Institute of Standard and Technology, (NIST). *Recommendation for Key Management*. 6.2. Protection Mechanism. 6.2.1. Protection Mechanism for Cryptographic Information in Transit: Availibility, Integrity, Confidentiality. B.3.14. Other Cryptographically Related Material. B.3.14.4. RNG Seeds. B.3.14.8. Random Numbers. Special Publication 800-57. MD 20899-8930. July 2012

[10] Bauhaus-Universit at Weimar, German. *Catena : A Memory-Consuming Password-Scrambling Framework*. Christian Forler, Stefan Lucks, and Jakob Wenzel

[11] Katholieke Universiteit Leuven, (KUL). *Cryptographic Algorithms - Research Challenges*. Stream Ciphers. Block Ciphers. Bart Preneel. Balkancrypt, Sofia. November 2013.

[12] Katholieke Universiteit Leuven, (KUL). *The MAKWA Password Hashing Function*. Specifications v1.0 Thomas Prnin February 22, 2014

[13] RFC2898, (RSA Laboratoires). *PKCS 5: Password-Based Cryptography Specification, Version 2.0*. 5. Key Derivation functions: PBKDF. p. 8-9. 7. Message Authentication Schemes: PBMAC. p. 16-17. B. Kaliski September 2000

[14] National Institute of Standard and Technology, (NIST). *Recommendation for Key Derivation Using Pseudorandom Functions*. 5. Key Derivation Functions (KDF). 5.1. KDF in counter mode 5.2. KDF in feedback mode 5.3. KDF in double pipeline iteration mode. Lily Chen. NIST Special Publication 800-108. October 2009.

[15] Springer Berlin Heidelberg *Towards Hardware-Intrinsic Security - Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions*. Foundations and Practice. p. 3-37 Springer Berlin Heidelberg Rainer Falk, Steffen Fries 2010

[16] SSpringer-Verlag Berlin Heidelberg *Hardware Intrinsic Security from Physically Unclonable Functions*. Helena Handschuh, Geert-Jan Schrijen, and Pim Tuyls 2011

[17] Munchen Germany *Optical PUFs Reloaded*. Ulrich Rührmair, Christian Hilgers, Sebastian Urban, Agnes Weiershäuser, Elias Dinter, Brigitte Forster, Christian Jirauschek. Germany 2013.

[18] Springer-Verlag Berlin Heidelberg *Physically Unclonable Functions: Concept and Constructions*. Chapter 2. R. Maes. Germany. DOI 10.1007/978-3-642-41395-7_2 2013.

[19] NCC Group *How to Backdoor Diffie-Hellman*. David Wong. NCC Group. June 2016

[20] Seoul National University (SNU), Republic of Korea *Lizard: Cut off the Tail! Practical Post-Quantum Public-Key Encryption from LWE and LWR*. Jung Hee Cheon, Duhyeong Kim, Joohee Lee, and Yongsoo Song. Korea.

[21] Laboratory for Computer Science, MIT, Cambridge, Department of Applied Science, Harvard University, Cambridge *Verifiable Random Functions*. Silvio Micali, Michael Rabiny and Salil Vadhanz.

[22] MIT *Aggregate Pseudorandom Functions and Connections to Learning*. Aloni Cohen, Shafi Goldwasser, Vinod Vaikuntanathan

[23] University of Germany, University of Belgium *PUFs: Myth, Fact or Busted? A Security Evaluation of Physically Unclonable Functions (PUFs) Cast in Silicon*. Stefan Katzenbeisser Ünal Kocabaş , Vladimir Rožić , Ahmad-Reza Sadeghi , Ingrid Verbauwhede and Christian Wachsmann. 2012.

[24] Bauhaus-Universitat Weimar, Germany *Catena: A Memory-Consuming Password-Scrambling Framework*. Christian Forler, Stefan Lucks, and Jakob Wenzel

[25] https://www.rc4nomore.com/

[26] https://www.tutorialspoint.com/cryptography/cryptography_hash_functions.htm

[27] http://www.slideserve.com/glenna/physical-unclonable-functions

[28] https://www.nature.com/articles/s41570-017-0031

[29] https://prezi.com/ocwebqegqclp/rc4-encryption/

[30] https://www.rc4nomore.com/

[31] http://www.cs.toronto.edu/ rackoff/2426f16/notes6.pdf

[32] https://cryptography.io/en/latest/hazmat/primitives/key-derivation-functions/

[33] https://www.allacronyms.com/KDF/Key_derivation_function

[34] http://cacr.uwaterloo.ca/hac/about/chap3.pdf

[35] https://security.stackexchange.com/questions/20803/how-does-ssl-tls-work

[36] http://www.openwall.com/presentations/PHDays2014-Yescrypt/PHDays2014-Yescrypt.pdf

[37] https://www.coursera.org/learn/hardware-security/lecture/Ab4sf/physical-unclonable-functions-puf-basics

[38] https://www.coursera.org/learn/crypto/lecture/A1ETP/key-derivation

[39] https://tools.ietf.org/html/rfc5869

[40] http://rijndael.ece.vt.edu/puf/background.html

[41] ieeexplore.ieee.org/document/6823677/

[42] http://people.csail.mit.edu/rudolph/Teaching/Lectures/Security/Lecture-Security-PUFs-2.pdf

[43] http://www.nhcue.edu.tw/ jinnliu/teaching/nde07/Lecture6.pdf

[44] https://web.mit.edu/kerberos/krb5-1.12/doc/admin/enctypes.html

[45] https://www.cryptolux.org/index.php/Argon2