

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 străinătate. 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 sursă al programelor și celelalte conținuturi (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 următoarele 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 licență "Generarea, derivarea, schimbul și managementul cheilor criptografice" reprezintă dorința de a studia în corelație domeniile "Criptografie" și "Securitatea informației".

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

Țin să mulțumesc domnului profesor Ferucio-Laurențiu Țiplea pentru întreaga coordonare, sprijinul și încurajarea oferită atât pe parcursul anilor de studiu cât și în demersul elaborării lucrării de licență.

Mulțumiri adresez și întregului colectiv de profesori cu care am avut deosebită plăcere să colaborez pe parcursul acestor trei ani de studiu, cât și tuturor colegilor, alături de care am reușit să elaborez proiecte, teme de laborator dar și alături de care anii de studenție au părut să fie unii dintre cei mai frumoși.

Cuprins

Declaratii standard	3
Introducere	6
1 Necesitate în contextul protocolului SSL/TLS	9
1.1 Argumentare	9
1.2 Inițiere în termeni criptografici și de securitate	10
1.2.1 Sistem de criptare / Criptosistem	10
1.2.2 Securitate	11
1.2.3 Criptografie simetrică	11
1.2.4 Criptografie asimetrică / cu chei publice	11
1.2.5 Securitate semantică	12
1.2.6 Securitate IND-CPA	12
1.2.7 Construcție scheme IND-CPA	12
1.2.8 Funcții hash	13
1.2.9 Număr random	14
1.3 Evoluția securității în timp	15
1.4 SSL/TLS - scurtă prezentare a protocolului	17
1.4.1 SSL&TLS: Autentificarea	18
1.4.2 SSL&TLS: Confidențialitatea	18
1.4.3 SSL Handshake	19
2 Generarea cheilor	20
2.1 Generarea cheilor simetrice	20
2.2 Generarea cheilor asimetrice	21
2.3 Generatori pseudo-random	22
2.4 Funcții pseudo-random	23
2.5 RC4 - Generator Pseudo Random	23
2.5.1 RC4:	24
3 Derivarea cheilor	28
3.1 Descriere generală a KDF(key derivation function)	28
3.2 PBKDF	31
3.3 HKDF	32
3.3.1 Generarea MAC-urilor	33
3.3.2 Verificarea MAC-urilor	33
3.4 ConcatKDFHash - ConcatKDFHMAC	34
3.5 Moduri ale KDF	35
3.5.1 Counter Mode KDF	35
3.5.2 Feedback Mode KDF	36
3.5.3 Double Pipeline Iteration Mode KDF	36
4 PUF - Physically Unclonable Functions	38
4.1 Tipurile principale de PUF-uri	39
4.1.1 PUF-uri optice	40
4.1.2 PUF-uri decalate	40

4.1.3	PUF-uri SRAM (Static Random Access Memory)	41
4.1.4	PUF-uri fluture	41
4.2	Aplicații ale PUF-urilor	42
4.2.1	Abordări anti-fraudă	42
Contribuții		44
Concluzii		46
Bibliografie		47

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 proprietăți 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, cât 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), $|M| > 1$, 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 $k_1, k_2, k_3, \dots, k_n$)
2. E este un algorithm 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 algorithm determinist de decriptare care, pornind de la un criptotext $c \in C$, aparținând mulțimii de criptotexte C , și o cheie $k \in K$, 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 $c \in C$ să fie obținut cu algoritmul de criptare care primește la intrare mesajul $m \in M$.

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

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

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

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 $k \in K$; $k < -G(\lambda)$.
- E este un algoritm probabilist de complexitate timp polinomială care, pornind de la o cheie $k \in K$, și un mesaj $m \in M$, generează un criptotext $c \in C$, $c < -E(k, m)$.
- D este un algoritm determinist de complexitate timp polinomială care, pornind de la o cheie $k \in K$ și un criptotext $c \in C$, generează un mesaj $m \in M$, $m < -D(k, c)$.

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

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 polinomială 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 $c \in C$ folosind cheia publică generată anterior de către generatorul G , $c \leftarrow E(pk, m)$.
- D este un algoritm determinist de complexitate timp polinomială care, aplicat criptotextului $c \in C$ împreună cu cheia secretă sk , descoperă mesajul inițial criptat $m \leftarrow D(sk, c)$

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 făcută publică 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 neglijabilă 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) : adversarul 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-CPA Encryption	Enc (ms)	Dec (ms)	Cpvt (bytes)
Lizard	0.010	0.020	745
CHK+	0.314	0.106	770
IND-CCA Encryption	Enc (ms)	Dec (ms)	Cpvt (bytes)
CCA-Lizard	0.012	0.025	778
CCA-CHK+	0.313	0.302	804

IND-CCA.

1.2.7 Construcție scheme IND-CPA

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

GENERATORI PSEUDO-RANDOM:

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

FUNCȚII PSEUDO-RANDOM:

O funcție pseudo-random este o familie de funcții indistingibilă față de familia tuturor funcțiilor. Un adversar care primește o funcție nu va putea distinge cu probabilitate ne-neglijabilă dacă funcția este aleasă din acea familie sau dacă este o funcție arbitrară aleasă random. În criptografie notăm funcțiile pseudo-random prin $F = (F_k)$ 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 adevărată următoarea afirmație:

Probabilitatea ca un adversar să câștige experimentul este neglijabilă.

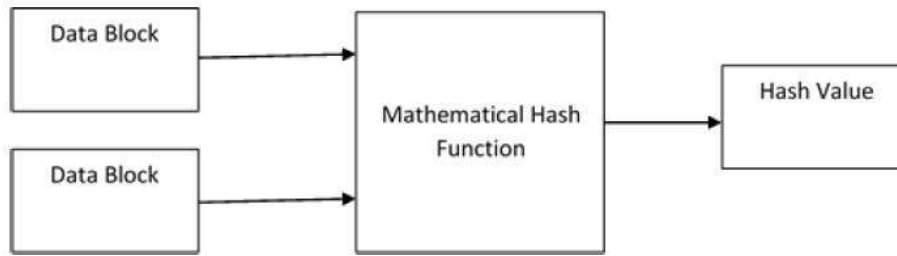
$| P(Priv_{A,S}^{CPA}(\lambda) = 1) - 1/2 |$ este funcție neglijabilă în λ unde $Priv_{A,S}^{CPA}(\lambda)$ este experimentul pentru adversarul A .

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 $\{0, 1\}^l$. $f : \{0, 1\}^* \rightarrow \{0, 1\}^l$ unde $l \geq 1$ este fixat.



Proprietăți ale funcțiilor hash:

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

1. Rezistența la coliziuni:

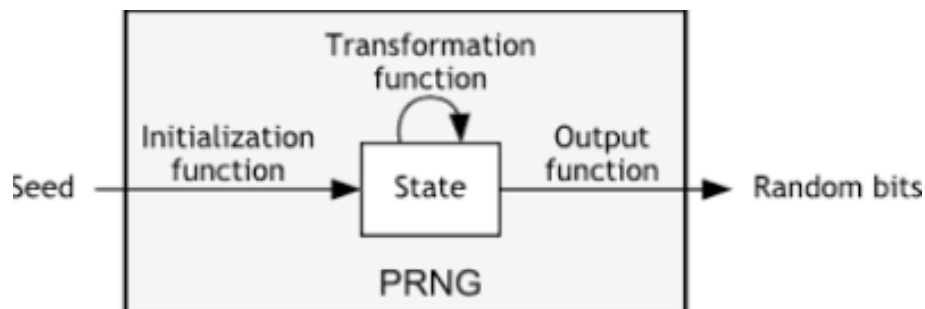
- Pentru un posibil atacator A este intractabil să determine două mesaje diferite și de lungimi diferite $m_1, m_2 \in \{0, 1\}^*$, $m_1 \neq m_2$ dacă funcția hash aplicată primului mesaj este identică cu funcția hash aplicată celui de-al doilea mesaj: $h(m_1) = h(m_2)$, (m_1, m_2) însemnând o coliziune a funcției hash.
- Unui atacator A , având mesajul m_1 îi este intractabil să determine mesajul $m_2 \neq m_1$ astfel încât $h(m_1, m_2) =$ 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 $y \in \{0, 1\}^l$ atacatorului îi este intractabil să determine x , $f(x) = y$.

1.2.9 Număr random

Domeniul Criptografiei și al Securității informatice, folosesc frecvent conceptul de număr 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_1, a_2, \dots, 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 Eveniment

1970	•	Bell La Padula gândește o politică de determinare al controlului accesului pentru domeniul militar.
1970	•	Rețelele Feistel \Rightarrow 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 <u>confidențialitate</u> și <u>izolarea datelor</u> .
1984	•	Modelul schematic ce se situează 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 <u>integritate</u> și <u>valabilitatea datelor</u> .
1996	•	Schneier pune accentul pe <u>nepredictibilitate</u> și <u>nereproductibilitate</u> 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 <u>non-repudiare</u> , <u>posesia datelor</u> și <u>autenticitatea datelor</u> .
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: <u>cyber security</u> , <u>transparența</u> , <u>eficientizarea costurilor</u> , <u>eficientizarea securizării</u> .
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, TLS_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 cât 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 autenticată

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

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

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.

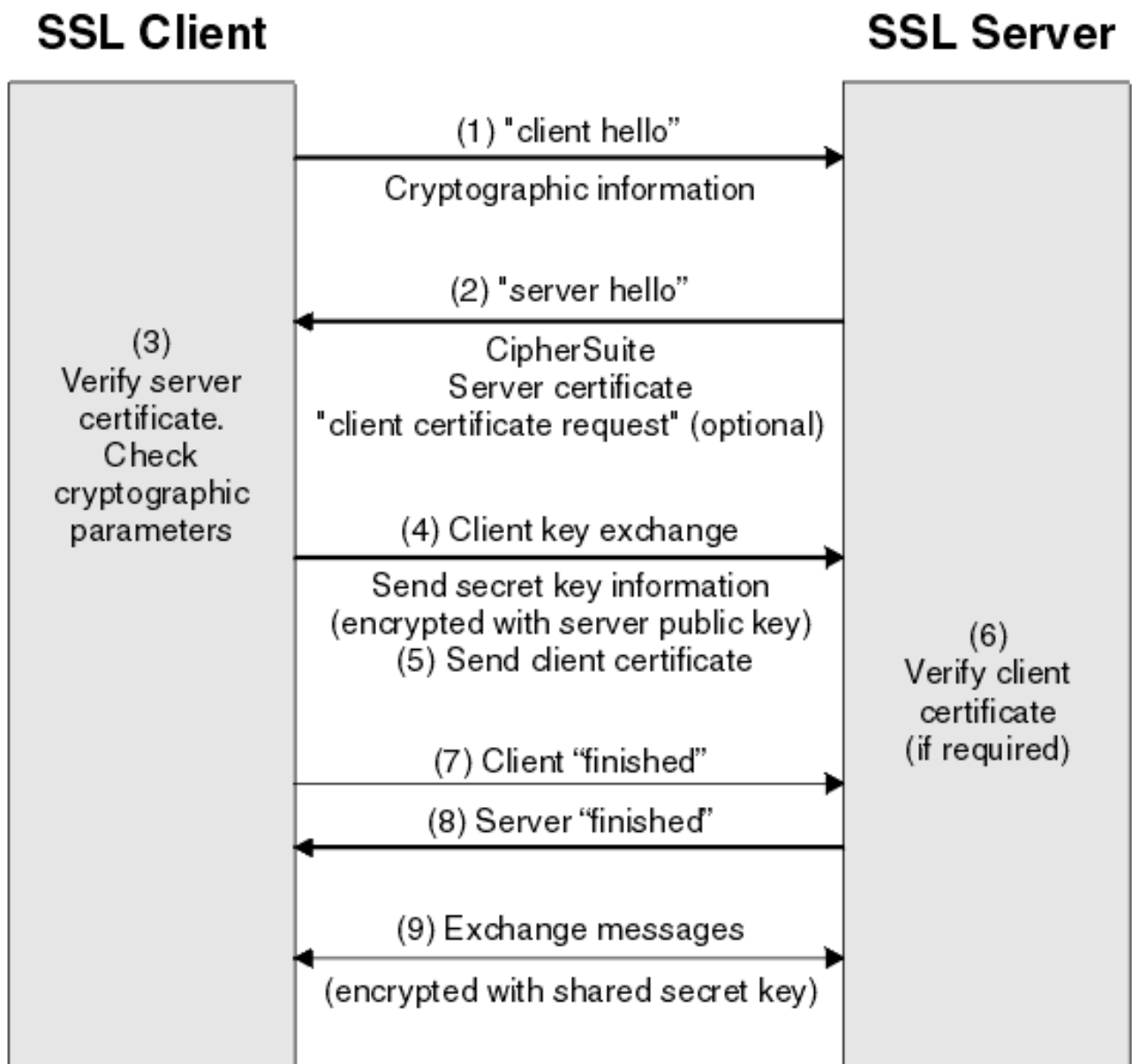
Schimbul de certificate digitale în timpul SSL/TLS handshake este o parte a procesului de autentificare.

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-repudiare etc.

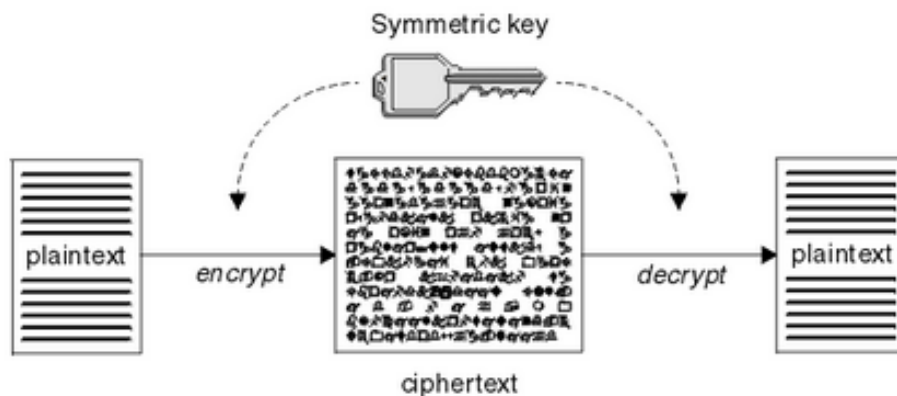
2.1 Generarea cheilor simetrice

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 privată $Priv_K$ 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



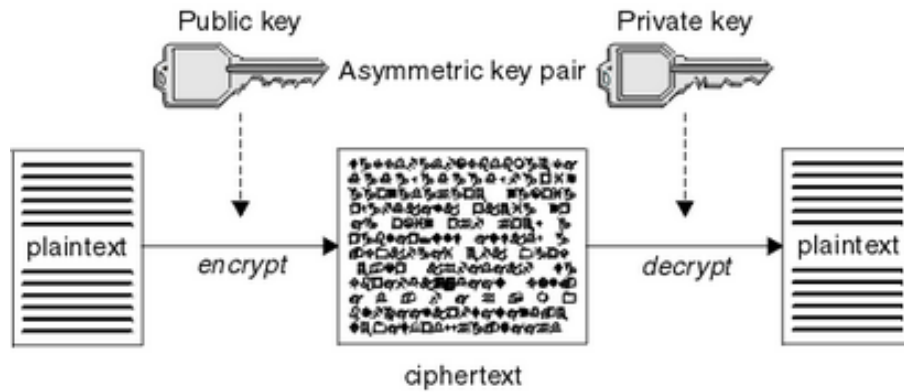
2.2 Generarea cheilor asimetrice

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, integritatea 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 integritatea.



După această scurtă introducere, prezentăm cititorului următoarele 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 , $s \in 0, 1^k$, generează șiruri w_i , $i \in [1, l(k)]$.

Generatorul trebuie să fie construit de așa natură încât unui posibil adversar A să îi fie indistingibil dacă $w_i \in 0, 1^k$ a fost generat de un generator PRG sau dacă a fost ales random uniform din mulțimea $\{0, 1\}^k$.

$0, 1^k \xrightarrow{\text{PRG}} w$
 $0, 1^{l(k)} \xrightarrow{\text{PRG}} w$
 $A(w) = ? \xrightarrow{\text{PRG}} 0, 1^k \xrightarrow{\text{PRG}} \text{PRG}$
 $A(w) = ? \xrightarrow{\text{PRG}} 0, 1^{l(k)} \xrightarrow{\text{PRG}} \text{random uniform}$

Definition 2.3.1. Fie $n \geq 1$ un număr natural ($\in \mathbb{N}$) și l un polinom ($\in \mathbb{P}$). 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. $l(n) \geq n, \forall n \in \mathbb{N}$
2. Pentru orice algoritm de complexitate timp polinomială PPT D , are loc și este neglijabilă următoarea 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 $0, 1^l(n)$.
- $D(G(s))=1$, înțelegem că D este un algoritm de test care decide dacă $G(s)$ este ales random uniform din $0, 1^l(n)$, $s \in 0, 1^n$.

2.4 Functii pseudo-random

În această secțiune propunem să facem cunoscut cititorului următoarele 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 și 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
 $F_k : 0, 1^n \rightarrow 0, 1^n$
 $m=m_1, m_2, \dots, m_l, |m_i|=n/4$
 l se scrie pe $n/4$ biti
 Tag-ul pentru m_i se obține prin : $r \leftarrow 0, 1^{pow(n/4)}, t_i = F_k(r \parallel 1 \parallel [i] \parallel m_i)$
 $t = (r, t_1, \dots, t_l)$ 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

$$m = ..m_1.....m_2 m_l$$

$$0..0 \rightarrow \oplus \rightarrow \oplus \rightarrow ... \rightarrow \oplus$$

$$..... \downarrow \downarrow \downarrow$$

$$..... t_1 \dots t_2 t_l$$

$$\text{MAC}(k,m)=t_l$$

2. X CBC-MAC:

$F=(F_k) \rightarrow$ o funcție PRF (pseudo-random)

3 chei: K_1, K_2 și 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 $t_l = F_k(t_{l-1} \oplus m_l \oplus k_3)$

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 și 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 încât 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

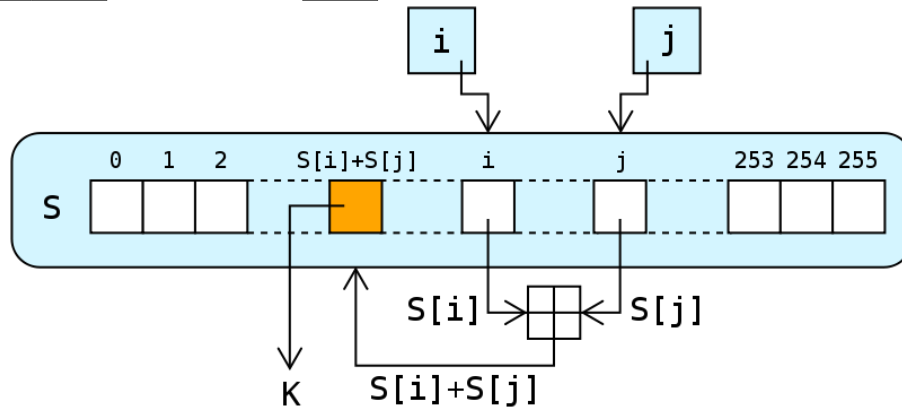
—output k

endwhile

CRIPTAREA :

..Plaintext.. \oplus ..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
→ $3 \text{ 0000 0011 } \oplus$
→ $\begin{array}{r} \text{---} \text{---} \text{---} \text{---} \\ 0100 \text{ 1011} \end{array}$
→ $i=1$, $j=1$, $S=2,1,3,0$
→ $i=i+1=2$
→ $j=i+S[i]=1+3 \pmod{4} = 0$
→ 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
→ $0000 \text{ 0001 } \oplus$
→ $\begin{array}{r} \text{---} \text{---} \text{---} \text{---} \\ 0100 \text{ 1000} \end{array}$

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 (~ 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 privește 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 exploatare cu o probabilitate ridicată.

Variante RC4 :

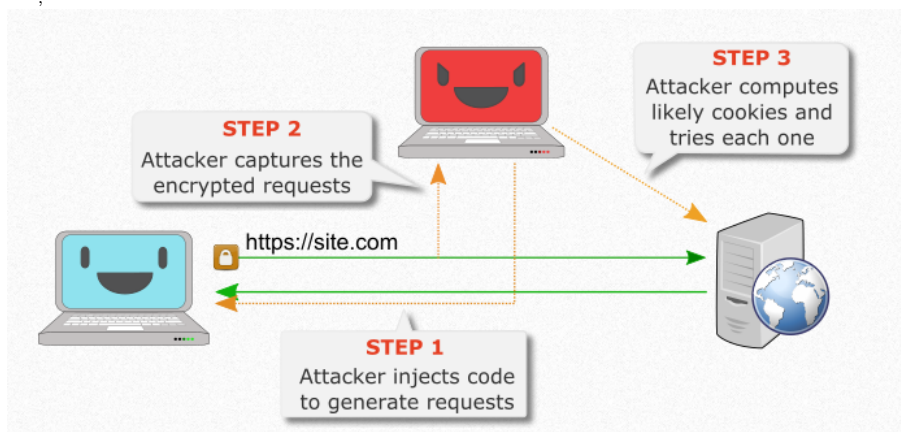
Diverse variante ale algoritmului au fost propuse, însă cele mai remarcate au fost RC4A (Varianta propusă de Souradyuti Paul și Bart Preneel, aduce doar o îmbunătățire a vitezei), RC4+ (key-schedule mai complex decât în RC4, mai greu de 1.7 ori decât RC4.) și Spritz (Ron Rivest și Jacob Schuldt, algoritmul 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 cercetărilor 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 decip-tând 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 $KDF = KeyDerivationFunction$ este o funcție pseudo-random cu ajutorul căreia una sau mai multe chei secrete (sk), aparținând 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, obținerea acestor chei într-un anumit format pentru a se potrivi protocoalelor de agreare de chei.

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

Introducem noțiunea de $KS = KeyStretching$ care este o tehnică de securizare a posibilelor chei slabe (parole sau passphrase) împotriva atacurilor brute-force (password cracking; funcțiile sunt în mod deliberat greu de calculat (ca timp) pentru prevenirea unor atacuri precum cel menționat anterior sau precum dictionary attack.), prin creșterea timpului de testare a tuturor posibilităților unei chei. Key strengthening extinde cheia cu un salt random și, spre deosebire de key stretching, șterge ulterior în mod sigur saltul adăugat.

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

1. DK = cheie derivată
2. KDF = funcție de derivare a cheii
3. Key = cheia inițială
4. $Salt$ = parametru ne-secret , număr random (utilizarea acestui parametru previne precalcularea unui dicționar de chei derivate [dictionary attack])
5. $Iterations$ = numărul de iterații al unei sub-funcții (dificultatea atacului crește proporțional cu numărul de iterații)

Cea mai importantă aplicație a KDF-urilor este **password hashing** -> verificarea parolelor, iar printre algoritmi de derivare a cheilor având și proprietăți de password hashing, care s-au remarcat, enumerăm pe următorii:

1. **Argon2** = câștigătorul competiției "Password Hashing Competition" 2015
Problemele existente înaintea apariției algoritmului Argon 2:

- Independente de input: în primele scheme, în care locația 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 să fie un bloc de memorie? Blocuri mici - încetinire din cauza principiului de cache CPU. Blocuri mari - greu de procesat din cauza numărului limitat de regiștri lungi.
- Dacă blocul de memorie este mare, cum să alegem funcția de compresie?
- Cum să exploatăm core-uri multiple de procesor?

Soluția Argon2: arhitectura x86 ce exploatează cache-ul și memoria procesoarelor Intel și AMD.

- Argon2d: mai rapid, utilizând accesul memoriei depinzând de date - perfect pentru aplicații fără amenințări din partea side-channel timing attack
- Argon2i: mai încet, efectuând mai multe operații pe memorie pentru a oferi securitate împotriva tradeoff attack.

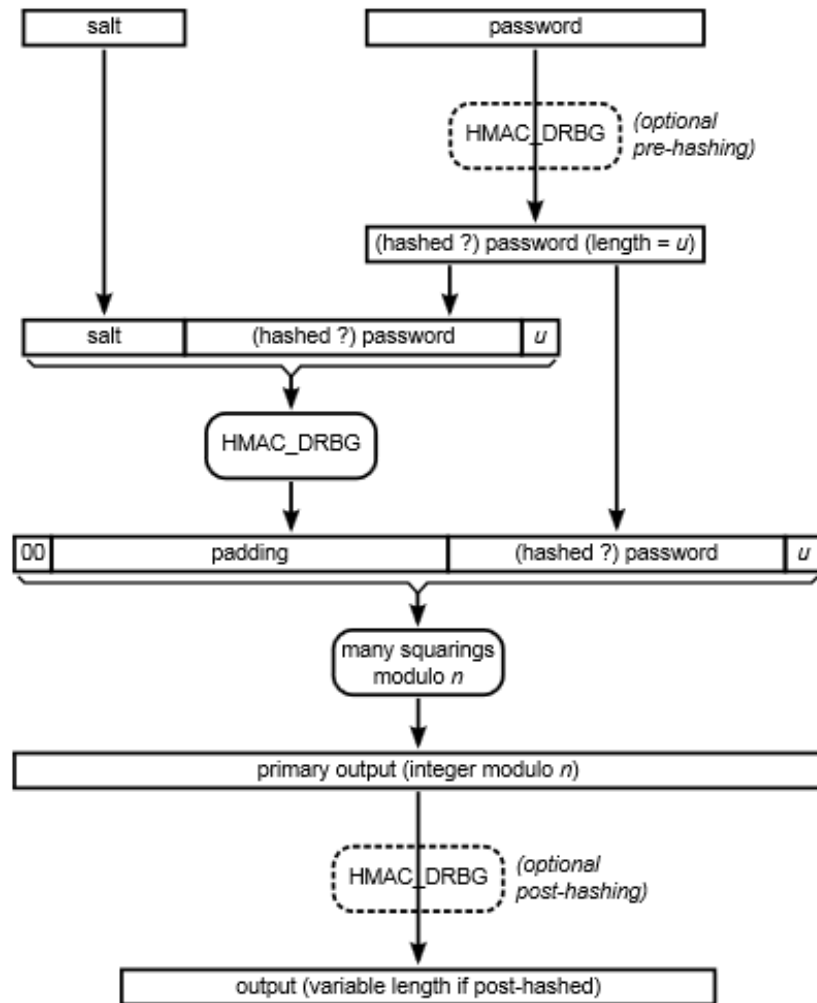
2. **Catena** = primul password scrambler care satisface atât consumul de memorie, prevăzând în același timp rezistența împotriva atacurilor cache-timing. Este un algoritm sigur, asigură indistingibilitate la numerele random și este rezistent la side-channel attack.

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]	$1-\infty$ (1,000)	small	-	-	(small memory)
bcrypt [44]	2^4-2^{99} ($2^6, 2^8$)	4,168 bytes	-	-	(constant memory)
scrypt [43]	$1-\infty$ ($2^{14}, 2^{20}$)	flexible, big	-	-	cache-timing attacks
CATENA-BRG (this paper)	$2^1-\infty$ ($2^{17}, 2^{20}$)	flexible, big	✓	✓	performance
CATENA-DBG (this paper)	$2^1-\infty$ ($2^{14}, 2^{16}$)	flexible, big	✓	✓	performance

Comparație a scramblelelor parolelor state-of-the-art și CATENA. Toți algoritmi menționați suportă valori de salt. Termenul $2^1 - \infty$ înseamnă factorul de cost care poate fi ales arbitrar de mare.

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

4. **Makwa** = transformă un input de lungime variabilă într-un output de lungime fixă. Oferă încetinire configurabilă dar și salt-ing. Diferit față de ceilalți 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 funcțiile necesare și se va încerca observarea datelor de către acest atacator pasiv, scopul fiind creșterea puterii computaționale a password hashing-ului.



Asupra parolei este aplicat opțional o funcție hash, după care se padează determinist (folosind salt-ul și funcția KDF). Asupra valorii finale se aplică apoi opțional o funcție hash.

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

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

Facem cunoscut cititorului faptul că o funcție 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 funcție KDF.

De asemenea, notăm faptul că puterea sau securitatea unei chei este măsurată de proprietatea de indistingibilitate față de adversar. Dat un adversar A , acestuia îi este imposibil a determina dacă output-ul a fost generat de o funcție KDF sau de o distribuție uniformă de string-uri de aceeași lungime de biți ca și KDF-ul, cu mențiunea că cheia de derivare a cheii este singura intrare necunoscută.

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

3.2 PBKDF

O aplicație ce derivează chei în modul Password-based KDF, trebuie să urmărească următorii pași:

1. Selectează un salt S și un counter de iterație c .
2. Selectează lungimea în octeți 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 = funcția de derivare a cheii
- PRF = funcție pseudo-random
- Password = parola din care se derivează o cheie
- Salt = secvența random de biți
- Iterations = numărul dorit de iterații
- dkLen = lungimea dorită a cheii derivate

$$DK = T1 \parallel T2 \parallel \dots \parallel T_{dkLen/hLen}$$

$$T_i = F(\text{Password}, \text{Salt}, \text{Iterations}, i)$$

$$F = \text{XOR}$$

Exemplu de derivare de cheie în protocolul WPA2 - Wifi-Protected Access:

$$DK = \text{PBKDF}(\text{HMACSHA1}, \text{passphrase}, \text{ssid}, 4096, 256)$$

PBKDF poate fi implementat cu un circuit mic și puțin RAM, ceea ce face ca un atac brute-force să fie ieftin, folosind circuite integrate sau unități de procesare grafică.

În urma competiției amintite și în secțiunea 3.1, "Password Hashing Competition", Argon2 a fost ales pentru derivare.

Diferențele dintre PBKDF1 și PBKDF2 se observă la nivelul funcțiilor aplicate. În timp ce prima aplică o funcție hash, cea de-a doua folosește o funcție pseudorandom. De asemenea, PBKDF1 produce chei ce nu sunt mereu suficiente ca lungime tuturor aplicațiilor, în timp ce PBKDF2 nemărginește lungimea cheilor, însă le limitează la structura funcției 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 integrității 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) și EAP-AKA (Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement).

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

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

$$\text{HKDF-Extract}(\text{salt}, \text{IKM}) \rightarrow \text{PRK}$$

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

$$\text{HKDF-Expand}(\text{PRK}, \text{info}, L) \rightarrow \text{OKM}$$

Aplicații 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 funcția hash Mercke-Damgard, construită pe o familie pseudo-random de funcții de compresie $\{hK\}$ K . Fie S o colecție de distribuții de probabilitate comportându-se ca o sursă de material de chei. Presupunând că instanțierea HMAC-ului cu familia de funcții de compresie prezentată mai sus este un extractor computațional sigur al sursei S , putem afirma că HKDF este o funcție de derivare a cheilor sigură în relație cu S .*

O funcție KDFHMAC este alcatuită, ca și după acronim, din două părți:

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

3.3.1 Generarea MAC-urilor

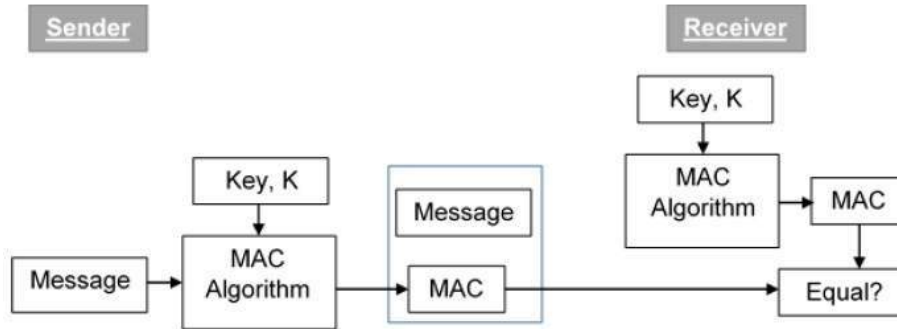
Generarea MAC-urilor produce un cod de autentificare a mesajelor dintr-un mesaj cu cheia K . Pașii necesari generării unui cod de autentificare a mesajelor sunt următorii:

1. Se stabilește S = saltul, c = counterul
2. Se stabilește lungimea cheii n octeți
3. Se aplică funcția de derivare a cheii $KDF(\text{parolă}, \text{salt}, \text{counter})D=DK$
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 . Pașii pentru certificarea codului de autentificare a mesajelor sunt următorii:

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



3.4 ConcatKDFHash - ConcatKDFHMAC

INPUT :

- Z : byte string ce reprezintă un secret partajat
- keydatalen : integer ce definește lungimea în biți a materialului secret de cheie ce trebuie să fie generat (ar trebui să fie $\leq \text{hashlen} * (2^{32} - 1)$)
- 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 și pentru ce algoritmi va fi folosit acest material.
 2. PartyUInfo = bit string conținând informații publice cerute de aplicație folosind KDF ce contribuie la un terț U la procesul de derivare a cheii.
 3. PartyVInfo = bit string conținând informații publice cerute de aplicație folosind KDF ce contribuie la un terț V în procesul de derivare a cheii
 4. SuppPubInfo = bit string conținând adițional informații publice cunoscute mutual.
 5. SuppPrivInfo = bit string conținând adițional informații 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 :

- $\text{reps} = \lceil \text{keydatalen} / \text{hashlen} \rceil$
- Dacă $\text{reps} \geq (2^{32} - 1)$, atunci abortează, afisează un indicator de eroare și oprește-te
- Inițializează un byte string de 32 biți în big-endian ca 00000001 în baza 16.
- Dacă $\text{counter} \parallel \text{Z} \parallel \text{OtherInfo}$ este $\geq \text{max_hash_inputlen}$, atunci abortează, afișează un indicator de eroare și oprește-te
- pentru $i=1$:
 - calculează $H_i = H(\text{counter} \parallel \text{Z} \parallel \text{OtherInfo})$
 - crește counter (modulo 2^{32}) tratându-l ca un integer pe 32 biți neasignat.
 - fie Hhas un set de Hash reps dacă $(\text{keydatalen} / \text{hashlen})$ este un integer; altfel, fie Hhash un set de $(\text{keydatalen} \bmod \text{hashlen})$ cei mai din stânga biți ai Hash reps.
 - setează $\text{DerivedKeyingMaterial} = H_1 \parallel H_2 \parallel \dots \parallel H_{\text{hashreps} - 1} \parallel H_{\text{hash}}$

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

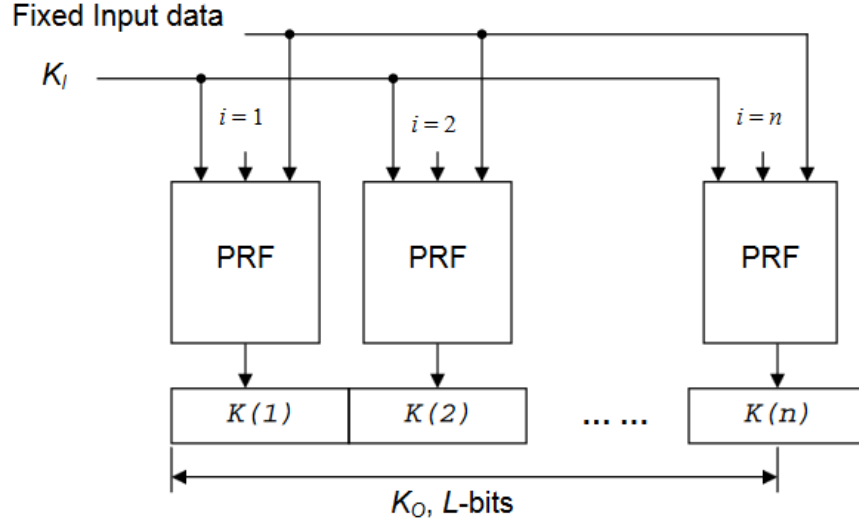
3.5 Moduri ale KDF

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

În funcție de modul de iterare peste funcția pseudorandom, identificăm trei tipuri de funcții KDF:

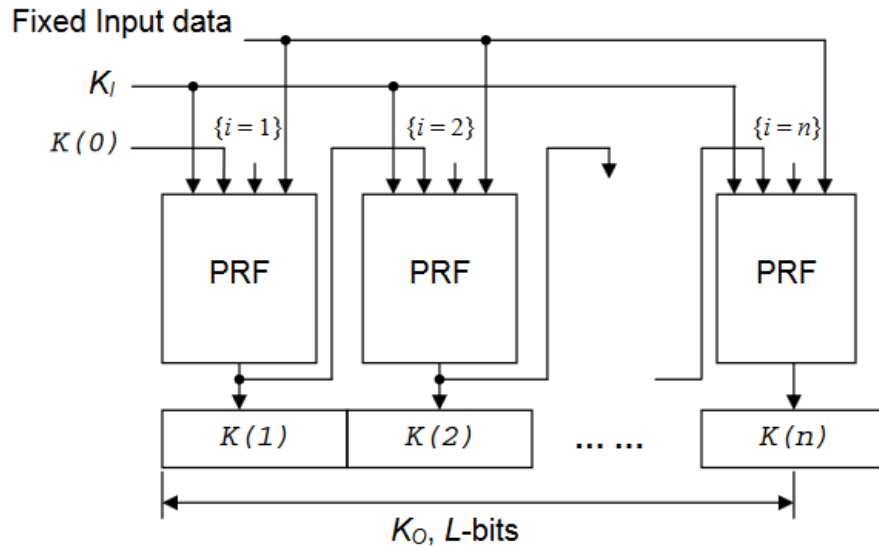
3.5.1 Counter Mode KDF

În modul counter, funcția pseudorandom este calculată cu un counter ca variabilă de iterație (Funcția pseudorandom este iterată în single pipeline).



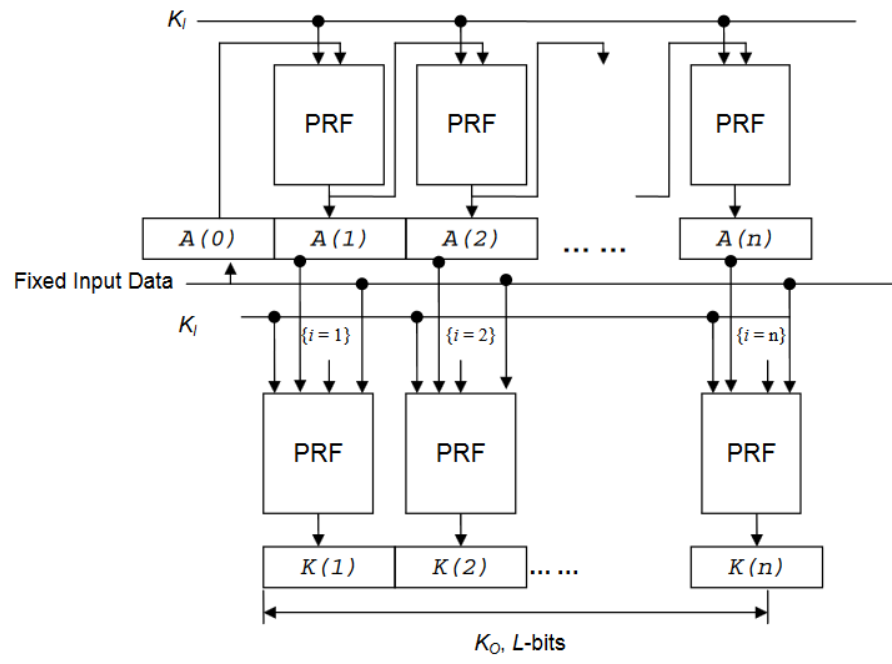
3.5.2 Feedback Mode KDF

În modul feedback, output-ul funcției pseudo-random este calculat folosind rezultatul iterației precedente și opțional, un counter ca variabilă de iterație. (Funcția pseudorandom este iterată în single pipeline).



3.5.3 Double Pipeline Iteration Mode KDF

Spre deosebire de modelele anterioare, funcția pseudo-random iterează în două pipeline-uri. Prima iterație generează o secvență de secrete folosite ca input pentru al doilea pipeline la același număr al iterației.



4. PUF - Physically Unclonable Functions

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

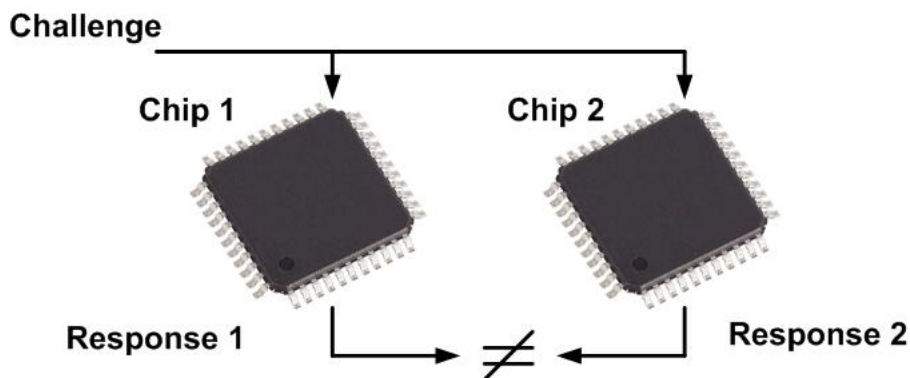
Capitolul prezent va face legătura cu domeniul Smart Card-urilor, relativ la PUF-uri, ce promit a asigura o "amprentă" prin generarea de chei criptografice unice fiecărui smart card în parte.

O funcție PUF trebuie să dețină următoarele proprietăți:

- Evaluabilă: ușor de evaluat.
- Unică: conține informații 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, funcția se transformă cu o probabilitate mare.

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

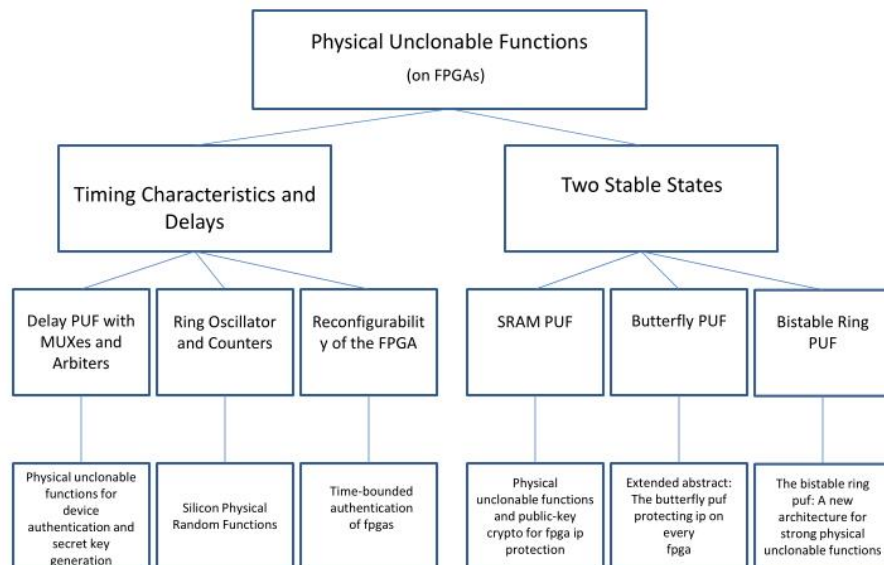
PUF-urile pot fi clasificate în două mari categorii, în funcție de randomitatea oferită:

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

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

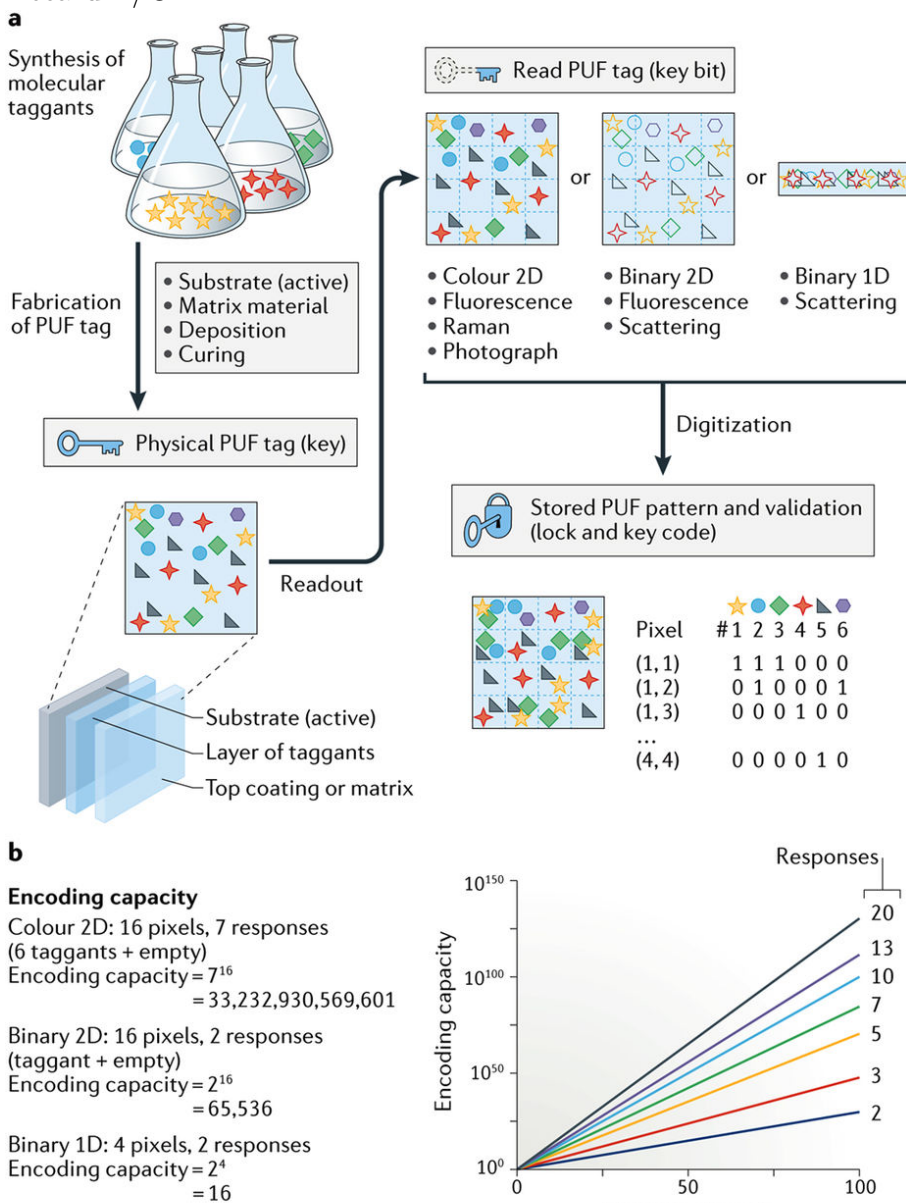
În cele ce urmează, vom descrie câteva dintre cele mai importante tipuri de funcții fizice neclonabile.



4.1.1 PUF-uri optice

PUF-urile optice sunt sisteme bazate pe fenomene optice ce dețin un mecanism care stabilește cu exactitate orice fenomen optic.

Se stabilește poziția relativă a token-ului de luminare a laserului și a camerei asupra fiecărui I/O.



Nature Reviews | Chemistry

4.1.2 PUF-uri decalate

PUF-urile decalate exploatează variațiile random ale firelor, cablurilor și porților din silicon. Având un challenge la intrare, circuitul pornește având două thread-uri sau tranziții ce propagă challenge-ul pe două căi diferite pentru a vedea pe care cale ajunge primul la final. Un "arbitru" analizează primul circuit terminat cu 1 și ultimul cu 0.

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

4.1.3 PUF-uri SRAM (Static Random Access Memory)

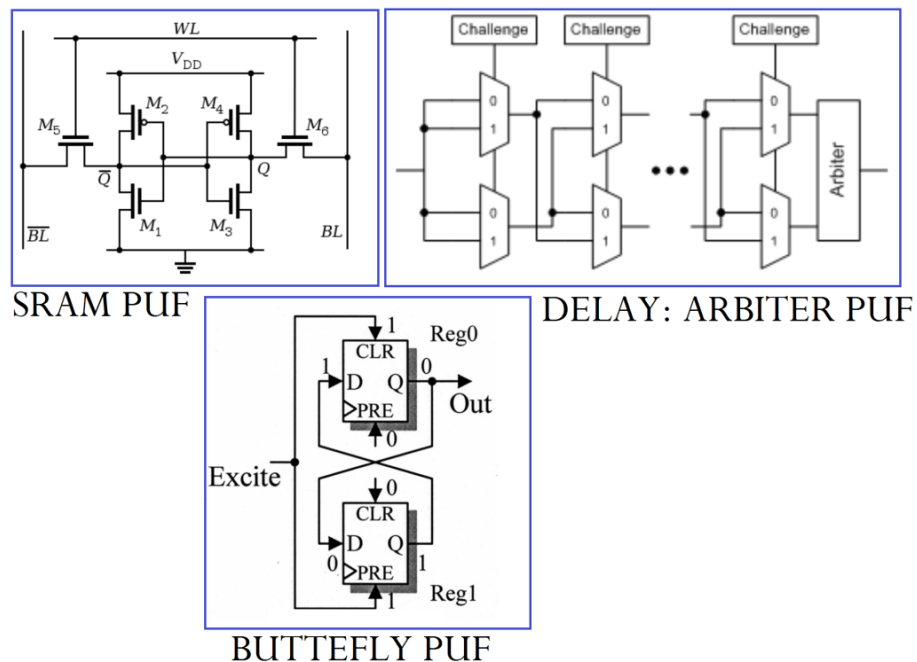
PUF-urile SRAM se află într-o cercetare continuă exploatându-se posibilele aplicații pentru prevenirea contrafacerii produselor sau a storage-ului sigur de chei criptografice.

Un exemplu în acest sens este tag-ul RFID (Radio-frequency Identification) [câmpuri electromagnetice pentru identificarea automată și tagarea produselor] ce poate fi foarte ușor clonat în absența unui PUF. În prezența 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 șir de celule PUF (formate din lacăte cuplate încrucișate) și un procesor.

Celula este provocată prin aducerea sistemului într-o stare instabilă, lăsând-o să convergă într-un interval de timp specific la o stare stabilă. Starea stabilă preferențială este determinată de diferența definită de variațiile procesului în timpul fabricației.



4.2 Aplicații ale PUF-urilor

Până în prezent, s-au remarcat deja foarte multe aplicații ale funcțiilor fizice neclonabile (PUF):

1. Extragerea de chei criptografice

- Folosește la protejarea împotriva execuției 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 funcționalitatea de autentificare.
- Un obiect sau circuit digital poate fi așadar 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 parții ce îl verifică și invers.

3. Marcarea produselor originale

- În China, Camerele foto Canon 60D DSLR folosesc un mecanism de marcarea a produselor originale pentru a le proteja împotriva contrafacerii.

Pe lângă aceste aplicații 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.

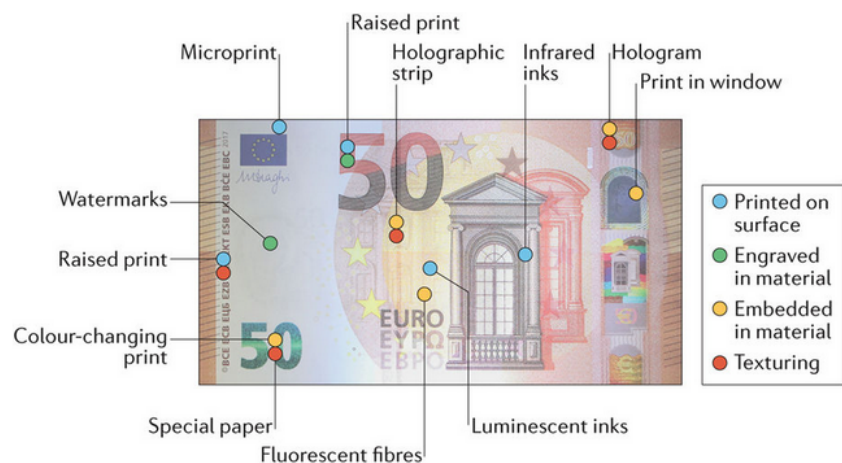
4.2.1 Abordări anti-fraudă

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

Legătura cu funcțiile 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

Soluția ar fi așadar introducerea unui tag anti-fraudă ce ar face o bancnotă imposibilă la clonare, precum funcțiile 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 luminescente, particule cu identitate, gravare cu laser sau control nanofotonic.

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

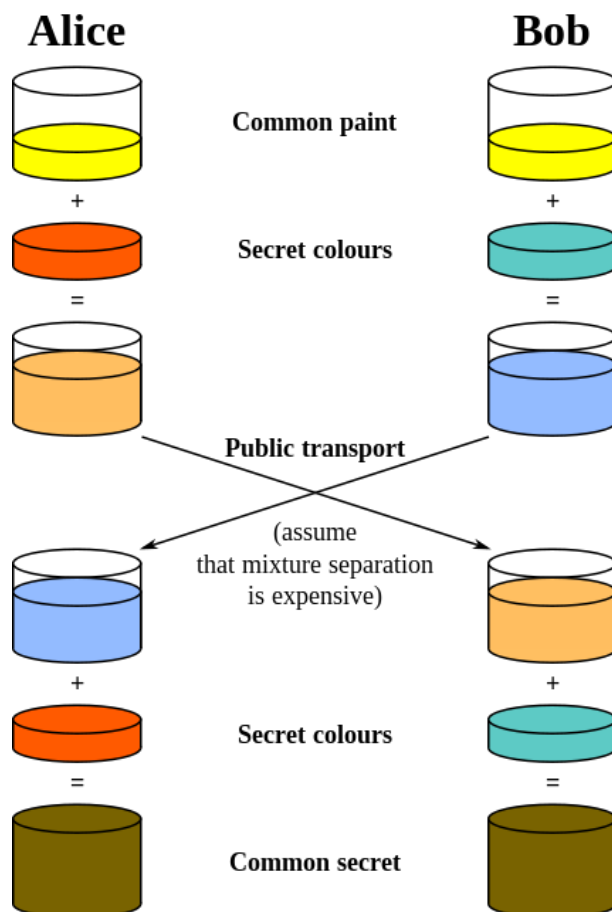
Contribuții

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

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

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

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



Pașii algoritmului Diffie-Hellman:

- Alice și Bob agreează folosirea modulului p în baza g (rădăcina primitivă modulo p)
- Alice alege un secret a și trimite lui Bob $A = g^a \bmod p$
- Bob alege un secret b și trimite lui Alice $B = g^b \bmod p$
- Alice calculează $s = B^a \bmod p$
- Bob calculează $s = A^b \bmod p$
- Alice și Bob au partajat secretul s

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

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

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

O soluție pentru a calcula aceasta ar fi metoda "trial multiplication" însă complexitatea se ridică la $q/2$ operații pentru a găsi soluția.

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} operații pentru găsirea soluției.

Cunoscând factorizarea completă a ordinului grupului, factorii fiind relativ mici, logaritmul discret poate fi calculat cu ușurință. 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 p_i la puterea k_i al ordinului.

O modalitate de a face aceste calcule este reducerea cheii secrete la subgrup. Calculând logaritmul discret al valorii y la puterea $(\phi p / (p_i \text{ la puterea } k_i))$, obținem cheia secretă $(\bmod p_i \text{ la puterea } k_i)$.

Valoarea obținută este un generator al subgrupului de ordin p_i la puterea k_i ridicat la puterea cheii secretă 1. Continuând și aplicând teorema chineză a resturilor, obținem cheia secretă finală.

Concluzii

Lucrarea de licență a adus o paralelă între Criptografie și Securitatea Informației, făcând referință la alte domenii precum Rețelele de Calculatoare și 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 nișe de interes ridicat.

Potențiale direcții viitoare de cercetare legată de tema actuală sunt dezvoltarea unui modul de integrare al celor mai importante atacuri în criptografie cât și cercetarea unor noi metode anti-fraudă deoarece acestea înglobează nu numai Informatica dar și Fizică și Chimie, științe 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 cryptographic keys: a critical component of cryptographic key management and the protection of sensitive information*. Cryptographic algorithms 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 derived from a pre-shared key. 7.5. Symmetric keys derived 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: Availability, 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-UNIVERSITÄT WEIMAR, GERMANY. *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 Vadhan.
- [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-UNIVERSITÄT 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-Yescript/PHDays2014-Yescript.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] <http://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/encetypes.html>
- [45] <https://www.cryptolux.org/index.php/Argon2>