

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V
BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

Evidenčné číslo: FEI-5384-64329

**LÚŠTENIE HISTORICKÝCH ŠIFIER NA GRIDE
DIPLOMOVÁ PRÁCA**

Študijný program: Aplikovaná informatika
Číslo študijného odboru: 2511
Názov študijného odboru: 9.2.9 Aplikovaná informatika
Školiace pracovisko: Ústav informatiky a matematiky
Vedúci záverečnej práce: Ing. Eugen Antal

Bratislava 2017

Martin Eliáš

Obsah

Úvod	1
1 Klasické šifry	2
1.1 História	2
1.2 Charakteristika	4
1.3 Útoky	5
1.3.1 Hrubou silou	5
1.3.2 Slovníkový útok	6
1.3.3 Genetické a evolučné algoritmy	6
2 Grid	7
2.1 hpc.stuba.sk	7
2.2 skripty PBS	10
2.2.1 Sériová úloha	10
2.2.2 Paralelná úloha	11
Záver	13
Zoznam použitej literatúry	14

Zoznam obrázkov a tabuliek

Zoznam skratiek

BASH	Bourne Again SHell
CPU	Central processing unit
GPU	Graphics processing unit
HDD	Harddisk drive
LAN	Local Area Network
PBS	Portable Batch System
RAM	Random Access Memory
SSH	Secure shell
VPN	Virtual private network

Zoznam výpisov

1	module avail	8
2	qstat	8
3	qstat -u 3xelas	9
4	qfree	9
5	uloha1.pbs	10
6	uloha2.pbs	11

Úvod

Tu bude krásny úvod s diakritikou atď.

A možno aj viac riadkový úvod.

1 Klasické šifry

V tejto kapitole sa budeme zaoberať históriou a stručným prehľadom klasických šifier. Spomenieme si aj niektoré základné útoky na klasické šifry.

1.1 História

História klasických šifier a utajovania písomného textu je pravdepodobne tak stará ako samotné písmo. Písmo, v podobe akej ho poznáme a používame dnes, pravdepodobne pochádza asi spred 3000 rokov pred Kristom a za jeho objaviteľov sa považujú Feničania. V niektorých prípadoch predstavovalo už použitie písma utajenie samotného textu. Príkladom môžu byť Egyptské hieroglyfy alebo klinové písmo používané v Mezopotámii. Iným príkladom môžu byť semitské jazyky, ktoré sú charakteristické používaním iba spoluhlások bez použitia samohlások, pretože tie zaviedli až Aremejci a po nich následné Gréci aby pomocou nich boli schopný rozlíšiť jazyky [1]. Aj diakritika ako taká má schopnosť rozlišovať významy slov, čo si ale až do 15. storočia nikto nevšimol, až pokiaľ ju Arabi nezačali používať pri kryptoanalýze rôznych šifier.

Z historického hľadiska nie je možné presne zoradiť ako jednotlivé šifry vznikali, pretože súčasne vznikali na viacerých miestach sveta. Komunikácia a s ňou spojené šírenie informácií nebolo také rýchle ako dnes, až do roku 1440 keď Johan Guttenberg vynašiel kníhtlač, čo zjednodušilo výmenu a uchovávanie informácií.

Ku kryptografii ako aj k rôznym iným vedným disciplínam prispelo v minulosti staré Grécko. Jedným z najvýznamnejších príspevkov starých Grékov bolo široké rozšírenie abecedy a písomného prejavu. Gréci písmo prebrali od Feničanov, ktorí na rozdiel od Egyptanov používali jednoduchšie písmo.

V Európe vďaka rozšíreniu abecedy začali vznikať aj prvé šifry, medzi ktoré patrí napríklad Cézarova šifra, ktorá vznikla v Rímskej ríši. Iným príkladom môže byť transpozičná šifra skytalé, ktorá bola používaná v Sparte.

Pád Rímskej ríše spôsobil úpadok kryptografie, ktorý trval až do obdobia stredoveku. Typickým znakom kryptografie v tomto období bolo napríklad písanie odzadu, alebo vertikálne, používanie cudzích jazykov, alebo vynechávanie samoh-

lások [1].

V stredoveku, kvôli bojom medzi pápežmi Ríma a Avignonu, bola kryptografia zdokonalená a začali sa používať rôzne kódy a nomenklátory. Ich charakteristickým znakom bolo zamieňanie písmen alebo nahradzovanie mien a titulov osôb v správach. V tomto období zabezpečovanie utajenia správ pokročilo až na takú úroveň, že na doručovanie správ boli použitý špeciálne vycvičení kuriéri.

V prvej polovici 20. storočia ľudia, ktorí pracovali v oblasti utajovanej komunikácie verili, že na to aby bola zabezpečená utajovaná komunikácia musí byť utajený kľúč a okrem neho aj šifrovací algoritmus. Toto ale odporovalo Kerckhoffovmu princípu, ktorý hovorí že: „Bezpečnosť šifrovacieho algoritmu musí závisieť výlučne na utajení kľúča a nie algoritmu“. Okrem toho sformuloval aj niekoľko požiadaviek na kryptografický systém, medzi ktoré patria:

1. systém musí byť teoreticky, alebo aspoň prakticky bezpečný
2. narušenie systému nesmie priniesť ťažkosti odosielateľovi a adresátovi
3. kľúč musí byť ľahko zapamätateľný a ľahko vymeniteľný
4. zašifrovaná správa musí byť prenášateľná telegrafom
5. šifrovacia pomôcka musí byť ľahko prenosná a ovládateľná jedinou osobou
6. systém musí byť jednoduchý, bez dlhého zoznamu pravidiel, nevyžadujúci nadmerné sústreďenie

Tieto princípy sú popísané v pôvodnej publikácii od Kerckhoffa [2].

Existovala ale aj iná skupina vedcov, medzi ktorých patrila aj Lester S. Hill, ktorý si uvedomoval že kryptológia je úzko spätá z matematikou. V roku 1917 si na Hillových prácach zakladal A. Adrian Albert, ktorý pochopil, že v šifrovaní je možné použiť viacero algebraických štruktúr. Neskôr toto všetko usporiadal a zdokonalil Claude E. Shannon, čo možno považovať za ukončenie éry klasických šifier [1].

1.2 Charakteristika

Na rozdiel od moderných šifier, ktoré sa používajú dnes, sú tie klasické rozdielne v niektorých hlavných črtách. Môžeme spomenúť niekoľko:

- Šifrovanie a dešifrovanie klasickej šifry možno realizovať zväčša pomocou papiera a ceruzky alebo nejakej mechanickej pomôcky.
- V dnešnej dobe aj vďaka rozšírenému použitiu počítačov stratila väčšina týchto algoritmov svoj význam.
- Utajuje sa algoritmus a aj kľúč a neuplatňuje sa Kerckhoffov princíp.
- Na rozdiel od moderných šifier sa používajú malé abecedy.
- V klasických šifrách je otvorený text, zašifrovaný text a kľúč v abecede reálneho jazyka, pričom v moderných šifrách sa používa binárne kódovanie.
- Na klasické šifry sa zväčša dá použiť štatistická analýza.

Z spomenutých charakteristík existujú aj výnimky. Napríklad pri Vigenereovej šifre sa algoritmus neutajoval. To platí aj pre Vernamovu šifru, ktorá okrem toho používa navyše binárne znaky. Vernamova šifra je perfektne bezpečná v podľa Shannonovej teórie [1].

Klasické šifry môžeme rozdeliť do niekoľkých základných kategórií:

- **Substitučné šifry.** V prípade že šifra permutuje znaky zdrojovej abecedy, hovoríme o monoalfabetickej šifre. Ako príklad môžeme uviesť šifru Atbaš prípadne Cézarovu šifru, alebo iné. V inom prípade ak sa aplikuje viacero permutácií podľa polohy znaku v otvorenom texte, tak hovoríme o polyalfabetickej šifre. Príkladom je Vigenerova šifra. Ďalším prípadom je polygramová šifra, kde sa z otvoreného textu najprv vytvoria bloky, na ktoré sa potom aplikuje nejaká permutácia.
- **Transpozičné šifry.** Transpozičné šifry sú vlastne blokové šifry, ktoré pri šifrovaní a dešifrovaní aplikujú pevne zvolenú permutáciu na každý blok ot-

voreného/zašifrovaného textu. Od polyalfabetickej šifry sa líši v poradí vykonávania operácii.

- **Homofónne šifry.** Homofónne šifry sú šifry, ktoré majú znáhodnený zašifrovaný text. Tieto šifry sa snažia zabrániť frekvenčnej analýze textu.
- **Substitučno-permutačné šifry.** Ak aplikujeme viacero substitučný a permutačných šifier na otvorený text tak hovoríme o substitučno-permutačných šifrách. Šifrovanie prebieha tak, že blok otvoreného textu sa rozdelí na menšie bloky, na ktoré je potom aplikovaná substitúcia, a permutácia, ktorá sa aplikuje na celý blok. Substitúcia zabezpečuje konfúziu a permutácia difúziu.

1.3 Útoky

1.3.1 Hrubou silou

Útok hrubou silou (bruteforce) je typ útoku, ktorý sa snaží zlomiť kľúč tak, že sa prehľadáva celý priestor kľúčov. Aby bol takýto útok možný a prakticky realizovateľný, priestor prehľadávaných kľúčov nesmie byť väčší ako hranica daná dostupnými prostriedkami alebo časom potrebným na riešenie.

Pre ilustráciu si uveďme jednoduchý príklad. Majme zašifrovaný text „VECDXSORS CDYBSMUIMRCSPSOBXKQBSNO“, ktorý vieme že bol zašifrovaný šifrou podobnou Cézarovkej šifre. Pre získanie otvoreného textu potrebujeme vyskúšať všetkých 26 možností posunov, čo je v tomto prípade kľúč, tak aby sme dostali zmysluplný text.

klúč 1

VECDXSORS CDYBSMUIMRCSPSOBXKQBSNO
WFDEPYTPSTDEZCTNVJNSDTQTPCYLRCTOP

klúč 2

VECDXSORS CDYBSMUIMRCSPSOBXKQBSNO
XGEFQZUQTUEFADUOWKOTEURUQDZMSDUPQ

klúč 3

VECDXSORS CDYBSMUIMRCSPSOBXKQBSNO
YHFGRAVRUVFGBEVPXLPUFVSVREANTEVQR

... // ďalšie kľúče 4..26

Po prezretí všetkých možností by sme zistili že kľúč 16 sa dešifruje na „LUSTENIEHISTORICKYCHSIFIERNAGRIDE“.

1.3.2 Slovníkový útok

Slovníkový útok narozdiel od útoku hrubou silou skúša iba niektoré možnosti z vopred pripraveného slovníka kľúčov.

Ukážme si ako by v princípe mohol fungovať slovníkový útok na šifru Vigenere. Nech zašifrovaný text je „SYKESUMWSWZXGCWJOQNVZMXTSYRSRFPHW“. Útočník má k dispozícii slovník slov „ABC, SOMAR, HESLO, ...“.

klúč JANO

SYKESUMWSWZXGCWJOQNVZMXTSYRSRFPHW
JYXQJUZI JWMJXCJVFQAHQMKFJYEEIFCTN

klúč SOMAR

SYKESUMWSWZXGCWJOQNVZMXTSYRSRFPHW
AKYEBCYKSFHJUCFRAENEHYLTBGDGROXTK

klúč HESLO

SYKESUMWSWZXGCWJOQNVZMXTSYRSRFPHW
LUSTENIEHISTORICKYCHSIFIERNAGRIDE

1.3.3 Genetické a evolučné algoritmy

todo

2 Grid

Jedným z cieľov práce je preskúmať možnosti aplikovania útokov na klasické šifry v gridovom prostredí. Grid môžeme chápať ako skupinu počítačov, uzlov, spojenú pomocou siete Local Area Network (LAN), prípadne inou sieťovou technológiou, ktoré môžu ale nemusia byť geograficky oddelené. Účelom takýchto počítačov je poskytnúť veľký výpočtový výkon, ktorý je použitý na riešenie špecifických úloh.

2.1 hpc.stuba.sk

V rámci Slovenskej technickej univerzity (STU), Centra výpočtovej techniky (CVT) sa nachádza superpočítač IBM iDataPlex, ktorý pozostáva z 52 výpočtových uzlov. Každý výpočtový uzol má nasledovnú konfiguráciu:

- CPU: 2 x 6 jadrový Intel Xeon X5670 2.93 GHz
- RAM: 48GB (24GB na procesor)
- HDD: 2TB 7200 RPM SATA
- GPU: 2 x NVIDIA Tesla M2050 448 cuda jadier, 3GB ECC RAM
- Operačný systém: Scientific Linux 6.4
- Sieťové pripojenie: 2 x 10Gb/s Ethernet

Spolu máme k dispozícii 624 CPU, 3584 cuda jadier, 2,5TB RAM, 104TB lokálneho úložného priestoru a ďalších 115TB zdieľaného úložiska. Výpočtový výkon dosahuje 6,76 TFLOPS a maximálny príkon aj spolu s chladením je 40kW [3].

Aby sme boli schopný grid používať musíme si najprv zaregistrovať projekt a požiadať o vytvorenie používateľského účtu na stránke výpočtového strediska **hpc.stuba.sk**. Po registrácii a získaní prihlasovacích údajov sa môžeme prihlásiť do webového rozhrania, cez ktoré môžeme spravovať projekt, pridávať ďalších riešiteľov, prezerať si štatistiky a grafy. Dôležitou funkciou webového rozhrania je zmena hesla používateľa a pridanie SSH verejného kľúča, pomocou ktorého sa môžeme prihlasovať bez zadávania hesla.

Do gridu sa môžeme prihlásiť cez SSH zadáním príkazu *ssh login@hpc.stuba.sk* a následným zadáním hesla v prípade ak nepoužívame prihlasovanie pomocou verejného kľúča. Ak sa pripájame mimo univerzitnej siete STU, na prihlásenie musíme použiť VPN. Po pripojení máme k dispozícii štandardnú linuxovú konzolu, ktorá ale obsahuje niekoľko špecifických príkazov pre daný grid. Zaujímať nás budú príkazy: *module*, *qstat*, *qfree*, *qsub*, *qsig*. Niektoré výstupy sú pre svoju obsiahlosť skrátené.

Príkaz *module* slúži na rýchle nastavenie ciest k vybraným knižniciam. Existujúce moduly môžeme vypísať pomocou *module avail*

/apps/modulefiles		
abyss/1.3.7	gaussian/g03	mvapich2/2.1
ansys/15.0	gaussian/g09	mvapich2/2.2
cmake/2.8.10.2	gcc/4.7.4(default)	nwchem/6.1.1(default)
cmake/3.1.0	gcc/4.8.4	nwchem/6.6
cp2k/2.5.1	gcc/4.9.3	openblas/0.2.18
cuda/6.5	gcc/5.4	openmpi/1.10.2
devel	gcc/6.3	openmpi/1.10.4
dirac/13.3	gridMathematica/9.0	openmpi/1.10.5
dirac/14	intel/composer_xe_2011	openmpi/1.4.5
esi/pamstamp	intel/composer_xe_2013	openmpi/1.6.5(default)
esi/pamstamp-platform	intel/libs_2011	openmpi/1.6.5-int8
esi/procast	intel/libs_2013	openmpi/1.7.2
esi/sysweld	matlab/R2015b	openmpi/1.7.5
fftw3/3.3.3	molcas/8.0	openmpi/1.8.8
fftw3/3.3.5	mvapich2/1.8a2	openmpi/1.8.8-int8
fftw3/intel-3.3.3	mvapich2/1.9(default)	openmpi/2.1.0
fluent/15.0.7	mvapich2/2.0	openmpi/intel-1.10.4

Výpis 1: module avail

Pre načítanie modulov zadáme *module load modul1 modul2 ...*, aktuálne používané moduly zobrazíme pomocou *module list* a odstrániť ich môžeme príkazom *module purge*. Podrobnejšie voľby príkazu *module* sa môžeme dozvedieť z manuálových stránok.

Ďalším dôležitým príkazom je *qstat*, ktorý zobrazuje status aktuálne bežiacich úloh. Detailnejší výpis o nami spustených úlohách môžeme vypísať cez *qstat -u \$USER* alebo *qstat -a*

Job ID	Name	User	Time Use	S Queue
--------	------	------	----------	---------

114557.one	halogen	3xjakubecj	499:03:0	R parallel
114640.one	JerMnchexFq5	3breza	218:35:9	R parallel
114663.one	Job4	3xrasova	78:07:20	R parallel
114668.one	run.opt	3antusek	674:08:1	R parallel
114692.one	Job5	3xbuchab	43:39:43	R parallel
114710.one	PGA	3xelias	226:46:1	R parallel

Výpis 2: qstat

Job ID	Queue	Jobname	SessID	TSK	Time	S	Time
114710.one	parallel	PGA	3418	96	120:00:00	R	19:08:38
115265.one	parallel	PGA_Mpi_3_b	24619	4	120:00:00	R	31:17:51
115266.one	parallel	PGA_Mpi_3_d	14748	4	120:00:00	R	31:17:51
115267.one	parallel	PGA_Mpi_3_e	14780	4	120:00:00	R	31:17:51
115268.one	parallel	PGA_Mpi_5_b	16429	6	120:00:00	R	31:17:50
115269.one	parallel	PGA_Mpi_5_d	16471	6	120:00:00	R	31:17:49
115270.one	parallel	PGA_Mpi_5_e	22492	6	120:00:00	R	31:15:43
115271.one	parallel	PGA_Mpi_5_f	22450	6	120:00:00	R	31:15:45
115272.one	parallel	PGA_Mpi_11_b	4254	12	120:00:00	R	31:12:39
115273.one	parallel	PGA_Mpi_11_d	—	12	120:00:00	Q	—
115274.one	parallel	PGA_Mpi_11_e	1647	12	120:00:00	R	31:12:08
115275.one	parallel	PGA_Mpi_11_f	22605	12	120:00:00	R	31:11:37

Výpis 3: qstat -u 3xelias

Posledný riadok tabuľky príkazu *qstat -u 3xelias* popisuje nami spustenú úlohu. Dôležité sú pre nás predovšetkým stĺpce **Time**, **Job ID**. Posledný stĺpec **Time** hovorí o tom ako dlho je už naša úloha spustená, druhý stĺpec **Time** nám deklaruje maximálny možný čas, ktorý má úloha PGA vyhradený. Hodnotu 114710 zo stĺpca **Job ID** môžeme použiť do príkazu *qsig* pre vynútené ukončenie úlohy.

Ak si chceme zobrazíť aktuálne vyťaženie gridu, môžeme tak urobiť príkazom *qfree*.

CLUSTER STATE SUMMARY						Local	GPFS Storage			
Core	1	...	12	load	FreeMem	Scratch	Read	Write	State	
Node	Queue				[GB]	[GB]	[MB/s]	[MB/s]		
comp01	S	[]	...	[]	0.00	44.44	0 (0.0%)	0.00	0.00	free
comp02	T	[]	...	[]	0.00	44.45	0 (0.0%)	0.00	0.00	free
...										
comp44	P	[O]	...	[O]	12.0	44.41	0 (0.0%)	0.00	0.00	job-exclusive
comp45	P	[O]	...	[O]	12.0	44.40	0 (0.0%)	0.00	0.00	job-exclusive
comp46	P	[O]	...	[O]	12.0	44.41	0 (0.0%)	0.00	0.00	job-exclusive

comp47	P	[O]	...	[O]	12.0	44.41	0 (0.0%)	0.00	0.00	job-exclusive
comp48	P	[X]	...	[X]	1.25	22.76	0 (0.0%)	45.50	2.83	job-exclusive
gpu1	G	[X]	...	[X]	7.98	38.33	0 (0.0%)	42.13	0.92	job-exclusive
gpu2	G	[]	...	[]	0.00	44.45	0 (0.0%)	0.00	0.00	free
gpu3	G	[]	...	[]	0.00	44.45	0 (0.0%)	0.00	0.00	free
gpu4	G	[]	...	[]	0.00	44.45	0 (0.0%)	0.00	0.00	free

Výpis 4: qfree

Prvý stĺpec popisuje názov výpočtového uzlu. Druhý stĺpec označuje druh fronty. S je pre sériové úlohy, T pre interaktívne úlohy, podobne P pre paralelné výpočty a G pre grafické výpočty. Stĺpce 1 až 12 značia procesory CPU respektíve GPU pre grafické výpočty. Zvyšné stĺpce ako už možno vyčítať z názvu popisujú celkové zaťaženie výpočtového uzla, voľnú pamäť a využitie diskov. Posledný stĺpec **State** popisuje či je daný uzol voľný alebo na vyťaženie. Procesory na ktorých prebieha výpočet našej úlohy sú označené ako „[O]“, zvyšné vyťažené CPU sú označené ako „[X]“, naopak voľné CPU sú označené medzerou „[]“ a v prípade že uzol nie je dostupný budú CPU označené ako „[-]“.

Posledný a najdôležitejší príkaz je *qsub*, ktorý slúži na zaradenie úloh, PBS skriptov do výpočtovej fronty.

2.2 skripty PBS

Aby sme boli schopný spustiť akúkoľvek výpočtovú úlohu na gride, potrebujeme k tomu Portable Batch System (PBS) súbor. PBS súbor je v skutočnosti iba jednoduchý textový súbor, ktorý definuje požiadavky na výpočtové zdroje a príkazy pre grid.

2.2.1 Sériová úloha

```

1 #!/bin/bash
2
3 #PBS -N uloha1
4 #PBS -l nodes=1:ppn=1
5 #PBS -l walltime=00:01:00
6 #PBS -A 3ANTAL-2016
7 #PBS -q serial
8
9 cd /work/3x Elias/uloha1
```

Výpis 5: `uloha1.pbs`

Vo výpise 5 môžeme vidieť príklad jednoduchého skriptu ktorý sériovej úlohy. Popíšme si jednotlivé riadky skriptu:

- Prvý riadok v súbore definuje aký shell sa má použiť pre spustenie skriptu. V našom prípade sme použili BASH, ale mohli by sme použiť aj iný shell alebo skriptovací jazyk.
- Tretí riadok určuje názov úlohy.
- Štvrtý riadok vymedzuje koľko uzlov a procesorov si žiadame od gridu. V tomto prípade si žiadame jeden výpočtový uzol a jeden procesor.
- Piaty riadok v PBS skripte vymedzuje aké časové rozpätie potrebujeme pre úlohu. V tomto prípade si žiadame jednu minútu.
- V šiestom riadku sa nachádza identifikátor podľa ktorého sa identifikujú úlohy s jednotlivými projektami. Tento parameter je povinný a možno ho získať po prihlásení na webový portál <https://www.hpc.stuba.sk/index.php?l=sk&page=login>.
- Parameter `-q` v siedmom riadku definuje typ výpočtovej fronty do akej bude úloha zaradená. V tomto prípade chceme úlohu zaradiť do fronty „serial“.
- V deviatom riadku sa presunieme do priečinku v ktorom sú uložené všetky potrebné dáta pre túto úlohu vrátane programu „seriova_uloha“.
- Posledný riadok spustí program „seriova_uloha“.

Úlohu môže zaradiť do fronty príkazom `qsub uloha1.pbs`.

2.2.2 Paralelná úloha

```
1 #!/bin/bash
2
3 #PBS -N paralelna_uloha
4 #PBS -l nodes=5:ppn=12
5 #PBS -l walltime=48:00:00
```



```

6 #PBS -A 3ANTAL-2016
7 #PBS -q parallel
8 #PBS -m ea
9 #PBS -M xelias@stuba.sk
10
11 . /etc/profile.d/modules.sh
12 module purge
13 module load gcc/5.4 openmpi/1.10.2
14
15 cd /work/3xelias/parallel
16 mpirun ./parallel

```

Výpis 6: uloha2.pbs

Podobne ako v predchádzajúcom príklade sériovej úlohy si popíšeme niektoré riadky príkladu `uloha2.pbs`:

- Na rozdiel od predchádzajúcej úlohy si v tomto príklade na riadku číslo štyri žiadame päť výpočtových uzlov a na každom uzle dvanásť CPU. Dokopy si žiadame šesťdesiat procesorov.
- V piatom riadku požadujeme časové rozpätie štyridsitich ôsmich hodín.
- Siedmy riadok definuje paralelnú frontu.
- Parametre `e` a `a` na riadku osem hovoria kedy sa má poslať email o zmene stavu úlohy. Parameter `e` znamená po skončení úlohy. Parameter `a` znamená pri zrušení úlohy. Ďalšie parametre môžu byť `b` (štart úlohy) a `n` (neposielat žiadny e-mail).
- V deviatom riadku definujeme e-mailovú adresu, na ktorú bude zaslaný mail v prípade ak úloha skončí alebo bude prerušená.

Záver

Conclusion is going to be where?

Here.

Zoznam použitej literatúry

1. GROŠEK, O., VOJVODA, M. a ZAJAC, P. *Klasické šifry*. Slovenská technická univerzita, 2007. ISBN 978-80-227-2653-5.
2. KERCKHOFFS, A. a COLLECTION, George Fabyan. *La cryptographie militaire, ou, Des chiffres usités en temps de guerre: avec un nouveau procédé de déchiffrement applicable aux systèmes à double clef*. Librairie militaire de L. Baudoin, 1883. Extrait du Journal des sciences militaires.
3. *STUBA klaster - IBM iDataPlex*. Dostupné tiež z: <https://www.hpc.stuba.sk>.