1999

Night Crypt

Šifrovací algoritmus

Jiří Altior
INSTITUT RENESANCE KULTURNÍHO DĚDICTVÍ

Algoritmus Night Crypt

Vznik a vývoj

V roce 1997 se mi naskytla možnost studovat Braillovo písmo To se skládá ze šestice bodů uspořádaných vedle sebe do dvou sloupečků po třech bodech. Kombinací zvýrazněných a nezvýrazněných bodů vznikají znaky, které mají přiřazený význam písmen, symbolů a číslic normální latinské, ale i řecké abecedy. Jelikož šestice bodů umožňují zapsat maximálně 64 znaků, jsou vyhrazeny speciální šestice, které fungují jako přesmykače. K ručnímu zápisu Braillova písma se používá speciálních tabulek, které obsahují šestice děr ve výše popsaném formátu. ty jsou uspořádány do řádků, přičemž každý řádek obsahuje konstantní počet šestic. Tvoří tedy vlastně matici bodů, do které jsou ony šestice zapisovány.

Když jsem později během řešení komplexní úlohy dynamického rozhraní byl postaven před nutnost zajistit bezpečnost souborů dat, napadlo mě použít principu zápisu Braillova písma na zašifrování dat. Během následujícího roku (tj. 1998) jsem se touto myšlenkou hlouběji zabýval, konzultoval jsem ji a diskutoval jsem o ní. Z počátku jsem ji vnímal jako zajímavou myšlenku, ale postupem času, jak jsem se dozvídal více informací o celé problematice kryptologie a kryptografie a jak se vyvíjela vlastní idea, došel jsem k přesvědčení, že algoritmus, který jsem navrhnul je jednou z rovnocenných možností, jak zabezpečit soukromí ukládaných dat a rozhodl jsem se tuto myšlenkou realizovat.

Jelikož jsem na algoritmu pracoval převážně v nočních hodinách, pojmenoval jsem jej Night Crypt. tato diplomová práce o algoritmu Night Crypt pojednává.

Poznámka k 1. PDF vydání

Předložený dokument je výňatek z diplomové práce zakladatele Institut renesance kulturního dědictví. Tuto práci úspěšně obhájil v roce 1999 na VTŠ (Vyšší technické škole) v Pardubicích. Tehdy se ještě jednalo o čtyřleté pomaturitní studium zakončené plnohodnotnou diplomovou prací. Autor byl oficiálně posledním absolventem, který dokončil úspěšně tuto formu vzdělávání v ČR a obdržel titul Diplomovaný technik. Tento titul se nesměl zkracovat a psal se za jménem. Nástupcem titulu Diplomovaný technik se stal titul DiS., který se vázal k úspěšnému završení studia na vyšší specializované škole (VOŠ). Inspirací k původnímu formátu studia (tzn. VTŠ) byl holandský model technického pomaturitního vzdělávání, které kladlo důraz na praktické zkušenosti studentů. Proto byla součástí vzdělávání VTŠ roční praxe, kterou autor absolvoval v pardubické pobočce společnosti PVT, a. s. Vedoucí jeho diplomové práce byl zaměstnaný právě v této společnosti. Oponentem této diplomové práce byl RNDr. Vlastimil Klíma z (tehdy plzeňské) společnosti Decros. Dnes je tento soudní znalec v oboru kybernetika považován za jednoho z nejpřednějších odborníků v kybernetice, který je celosvětově uznávaným specialistou na kryptologii.

Šifrovací algoritmu Night Crypt vychází z písma určeného pro slepce. Jeho ideou je práce různě uspořádanými n-ticemi bitů vkládaných do matice vyplněné statisticky upravenými daty ovlivněnými (zamíchanými) podle veřejného klíče. Finální aplikace šifrovacího mechanismu využívá principu morfování dvou matic, přičemž samotný proces morfování je ovlivňován proudem šifrovaných dat. Data, která jsou ve formě zašifrovaného textu výstupem obsahují stále stejný počet jedniček a nul, jako původní matice neovlivněná procesem šifrování. Pouze jejich uspořádání je pozměněno. Celý mechanismus pracuje tak, že se po vložení veřejného klíče začnou „míchat“ statisticky vyvážená data. Ta jsou zpracovávána v tzv. „stínové matici“. Na základě soukromého klíče je ten stejný proces na těch stejných datech realizován ještě jednou, ale tentokráte v pozměněné podobě, přičemž za ono „pozměnění“ je zodpovědný klíč a matematické důsledky z jeho užití. Třetím mechanismem, kde je v rámci šifrování použit algoritmus Night Crypt, je výsledná matice, jejímž vstupem je proud korelací, který vzniká procesem morfování šifrované matice do „tvaru“ stínové matice. Šifra zpracovává velikostně omezené bloky dat, které za sebe skládá. Celý proces šifrování si lze představit jako míchání různobarevných kuliček ve dvou stejných hrncích, přičemž jedno míchání se od druhého odlišuje daty ovlivněnými klíčem. Původní rozložení kuliček v hrncích je ovlivněno veřejným klíčem. Při míchání kuliček v obou hrncích je k nim přistaven ještě třetí hrnec, který také míchá kuličky, jenže tentokráte do nich přimíchává informaci (mění barvu kuliček) podle klíče, který vychází z toho, že se po různě dlouhých sekvencích míchání zastavuje míchání a porovnávají se stavy (rozmístění) kuliček v prvních dvou hrncích a faktorem ovlivňujícím míchání ve třetím výsledném hrnci jsou změny mezi „uspořádáními“ kuliček v obou hrncích (derivace stavu hrnců v čase zastavení míchání). Součástí procesu míchání jsou nejen měnící se rozmístění kuliček, ale i transformace parametrů matic (hrnců) či vnitřního uspořádání kuliček.

Jiří Altior
V Praze, 14. března 2022

Základní princip

Buňka

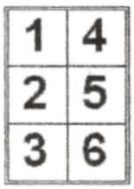
Šifrovací metoda Night Crypt pracuje na principu postupného vyplňování matice **n**-ticemi bitů. Metodu lze pro obraznost přirovnat k růstu buněk v organismu. Prázdná matice je postupně zaplňována jednotlivými buňkami, které se připojují k již existujícím. Algoritmus „bujení“ zaručuje, že se nemůže stát, aby jedna buňka zasahovala do druhé a v případě, že již neexistuje místo pro další buňku, umí obsah buněk doplnit do zbývajícího volného místa v matici.

Teoreticky může být buňka libovolnou **n**-maticí bitů, přičemž **n** musí být větší nebo rovno 1 a menší nebo rovno velikosti matice, ve které je **n**-tice použita. Z praktického hlediska musí být velikost buňky určená tak, aby splňovala kritéria bezpečnosti této šifrovací metody a aby se optimálně hodila k použitému algoritmu „bujení“. Velikost, tvar i vnitřní struktura buňky se může během algoritmu měnit a tím lze docílit bezpečnosti šifrovací metody (na úkor rychlosti zpracování).

Jako konkrétní příklad budu dále popisovat buňku o velikosti šest bitů, čili n-tici, kde **n=6**. Tato buňka je zobrazena na obr. 1.

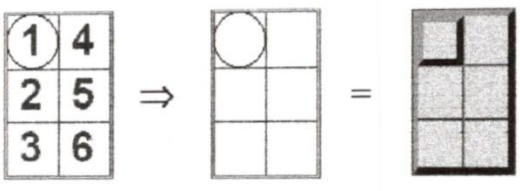
Obrázek 1: Buňka (6 bitů)

Jelikož zpravidla buňky obsahují více jak jeden bit, je vhodné jednotlivé bity v buňce vzestupně očíslovat, čímž jim zároveň bude přiřazeno pořadí. Na obr. 2 je zobrazena buňka o šesti bitech rozdělených do dvou sloupců po třech.

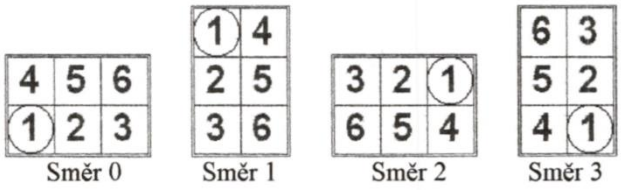


Obrázek 2: Rozložení bitů v buňce

Takto popsanou buňku lze zapsat do matice pouze jediným znázorněným způsobem. aby bylo možné buňku orientovat v prostoru matice (v našem případě se jedná o dvourozměrný prostor E2 reprezentovaný dvourozměrnou maticí), je zapotřebí ještě stanovit formát zápisu a schématického znázornění. Nejsnazším způsobem je v našem případě zvýraznit první bit v buňce (obr. 3) a směr orientace buňky v matici určit podle obr. 4.



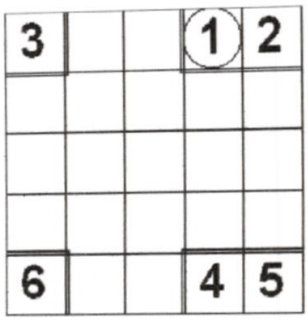
Obrázek 3: Označení prvního bitu a schématické označení



Obrázek 4: Směry orientace buňky v matici

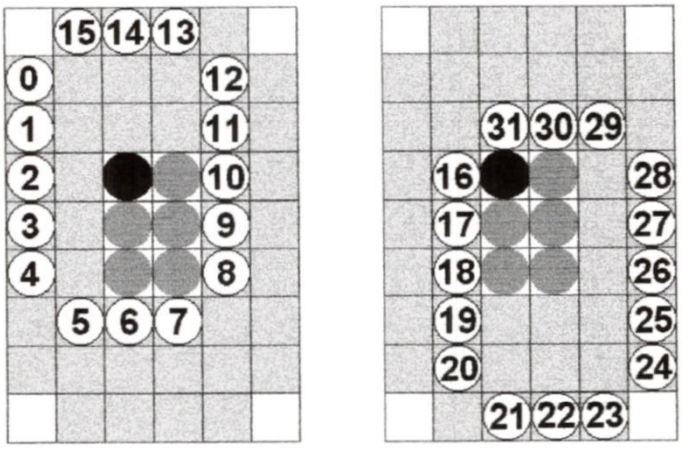
Algoritmus růstu buněk – bujení

Ještě než bude vysvětlena činnost algoritmu zajišťujícího zaplňování matice buňkami, je nutné definovat prostředí, ve kterém bude k „bujení“ docházet. Ve výše zmíněném případě se bude jednat o dvourozměrnou matici bitů čtvercového nebo přibližně čtvercového charakteru. Velikost matice bude maximálně 255*255 bitů (65025 b). Matice je tzv. uzavřená, což znamená, že první bit **x**-tého řádku má levého souseda (předchůdce) poslední bit téhož řádku, a naopak poslední bit **x**-tého má pravého souseda (následovníka) první bit **x**-tého řádku. Analogicky stejná vlastnost platí i pro sloupce matice. Buňku lze tedy umístit například jako na obr. 5.



Obrázek 5: Možné umístění buňky v matici

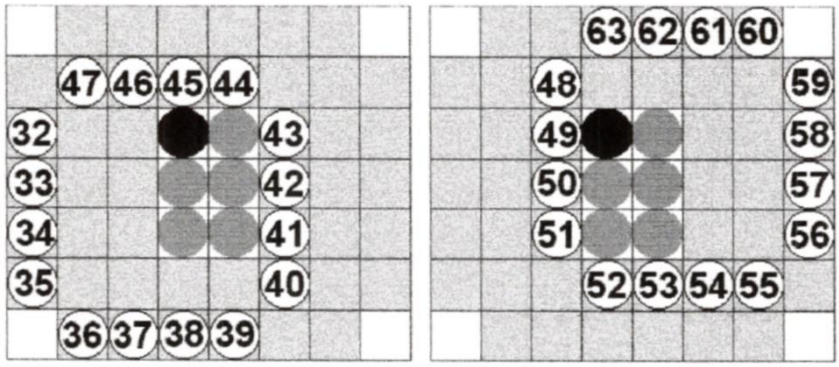
Základním pravidlem růstu buněk je, že nová buňka vzniká vždy vedle buňky již existující (výjimkou je první buňka v matice). To znamená, že minimálně jeden bit nové buňky a jeden bit již existující buňky spolu přímo jednou stranou sousedí. u zvolené vzorové buňky, která obsahuje šest bitů je celkem deset vnějších stran, kterými může sousedit s okolními buňkami. Jelikož jedna buňka může býti orientována čtyřmi různými směry (směr 0 až 3) existuje 64 možností, jak se původní buňka může rozrůst. Obecně platí, že počet vnějších stran bitů buňky je závislý na počtu vlastních bitů, ale i na prostorovém uspořádání buňky (přirozeně pro tři a více rozměrný prostor bude možností růstu při konstantním počtu bitů buňky přibývat).



Obrázek 6: Možnosti růstu buňky (směr 1) na buňku se směrem 1 a 3.

Na obrázku číslo 6 je schematicky znázorněna možnost růstu buňky s orientací ve směru 1 na buňku, která má orientaci 1 a 3. Jak bylo již výše zmíněno, nová i existující buňka může být orientována celkem čtyřmi způsoby podle směru vlastní orientace. To znamená, že 16 možných kombinací pro každý ze čtyř směrů orientace výchozí buňky krát 4 možné směry orientace nové buňky umožňuje celkem 256 vzájemných možných poloh dvou buněk. Všechny možné kombinace růstu jedné buňky jsou schematicky znázorněny v příloze 1.

Není-li volné místo pro uložení nové buňky na požadované pozici v matici, je nalezena první volná pozice v příslušném směru, kam jde novou buňku zapsat. Mechanismu vyhledávání volného místa je věnována následující kapitola **Zápis nových buněk do matice**. Za povšimnutí stojí, že interval 1..64 neboli 0..63 lze vyjádřit v binárním zápisu šesti bity, což je přesně počet bitů v jedné buňce. Tento fakt bude zmíněn v kapitole **Nesená a přidaná informace**.



Obrázek 7: Zbylé možné způsoby růstu buňky se směrem 1 na směr 0 a 2.

Zápis nových buněk do matice

Při vyhledávání volné pozice je zapotřebí nejdříve zjistit, zdali již nejsou obsazené bity, která nová buňka zabere. Jeli místo v matici ve zvolených bitech volné, je buňka na nové místo uložena a hledání je ukončeno. Zjistí-li se, že místo prázdné není, snaží se algoritmus vyhledat první volnou pozici pro buňku ve směru orientace buňky od výchozího místa po celé šířce či výšce matice. V případě, že je volné místo nalezeno, buňka se na nalezenou pozici uloží a algoritmus vyhledávání je ukončen. Může se ale stát, že se v přímém směru vyhledávání žádné volné místo nenajde. Potom se výchozí bit hledání volné pozice posune na sousední bit ve směru podle vzorečku:

$$\text{Směr} = (\text{Původní\_směr} + 1) \bmod 4. \quad (1)$$

Následuje opětovné vyhledávání volného místa pro buňku po celé šířce či výšce matice. Není-li nalezeno prázdné místo, výchozí bod vyhledávání se opět posune podle rovnice (1) a vyhledávání pokračuje, dokud není volné místo nalezeno. V případě, že je takto prohledána celá matice a není nalezené žádné volné místo pro uložení buňky v dotyčném směru, změní se orientace nové buňky podle vzorečku (1) a celý proces se opakuje s tím rozdílem, že pokud se odehrával zpočátku na sloupcích matice (směr 1 nebo 3) tak nyní se odehrává na řádkách (směr 0 a 2) a opačně. V případě, že nebude nalezeno ani nyní potřebné volné místo, algoritmus začne systematicky vyplňovat neobsazené bity v matici. Tímto způsobem je zaručeno úplné zaplnění matice.

V příloze 2 je ve čtyřech krocích názorně ukázán růst buněk v matici o rozměrech 5 řádků (A,B,C,D,E) po 18 sloupcích (1. až 18. sloupec bitů). V první části je schematicky znázorněna matice s jedinou buňkou orientovanou ve směru 0. V dalším kroku se k první buňce přidá buňka druhá ve směru 1, tj. rozroste se třicátým druhým možným způsobem (viz. příloha 1). Ve třetím kroku potom třetí buňka s orientací ve směru 2 čtyřicátým devátým možným způsobem a dále ve čtvrtém kroku buňka s orientací ve směru 3 padesátým prvním možným způsobem. Tato buňka se nachází zčásti na prvním a zčásti na posledním 18. sloupci matice.

Nesená a přidaná informace

Až dosud byl popisován mechanismus práce s buňkami, aniž bylo přihlíženo k jejich obsahu a bylo vysvětleno, jakým způsobem se určuje směr jejich dalšího růstu. Jak bylo zmíněno v kapitole 3.2.2, jedna zde popisovaná buňka se může rozrůst 64 možnými způsoby, což znamená, že je zapotřebí pracovat navíc ještě s informací o velikosti šesti bitů, která určí směr dalšího růstu buňky. V tomto konkrétním případě obsahuje jedna buňka šest bitů, takže je teoreticky možné ukládat informaci o dalším směru růstu bujení přímo do vlastní buňky. Takto by ale nezbylo žádné volné místo na šifrovaná data. Proto je zapotřebí najít vhodný způsob, jak zajistit, aby šifrovaná data byla v matici obsažena a zároveň byl určen směr nových buněk.

Bezpečnost kryptografického algoritmu Night Crypt je mimo jiné založena na faktu, že bez použití hrubé síly nelze bez znalosti klíče určujícího počáteční orientaci a pozici buněk určit, odvodit, vystopovat či jinak předvídat v aktuální případě konkrétní buněčný růst. Aby bylo dosaženo splnění této podmínky, je zapotřebí zajistit systematický růst buněk. Toho lze docílit několika metodami anebo jejich vzájemnou kombinací.

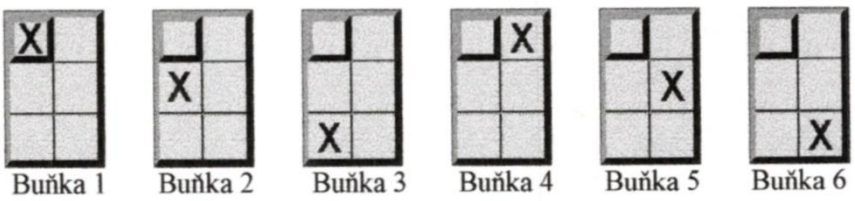
V nejjednodušším případě lze využít k určení dalšího kroku růstu buňky informaci, která je v samostatné buňce obsaženo, anebo v lepším případě lze směr nějakým způsobem vypočítat z hodnot předchozích **k** buněk, přičemž čím je **k** větší, tím je zajištěna menší pravděpodobnost, že bude následující růst předpovězen. velikou výhodou této metody je, že všechny bity obsažené v matici jsou šifrovanými daty a tudíž, že nedochází (až na výjimku dorovnání počtu bitů s počtem bitů matice) ke zvětšení velikosti zašifrovaných dat. Bezpečnost této metody ale závisí na kvalitě použitého výpočtu a mimo jiné i na vlastních šifrovaných datech. Moderní kryptoanalýza často využívá statistické metody, které na základě pravděpodobnosti a výskytu charakteristických znaků či rysů předpokládaného utajovaného sdělení dokážou rozluštit, nebo alespoň blíže specifikovat zašifrovaná data.

Druhou možností je použít k určení směru bujení náhodných čísel. Zde ale zákonitě dochází k prodloužení zdrojového textu určeného k zašifrování o bity reprezentující náhodně vygenerovanou posloupnost čísel. Vliv této nežádoucí vlastnosti lze sice vhodným návrhem snížit, ale nikdy mu nepůjde úplně zamezit. Slabým místem se také může stát samotný generátor náhodných (či pseudonáhodných) čísel.

Při použití právě zmíněné metody se stává aktuální nalezení vhodného kompromisu mezi objemem přidané informace a bezpečností celého systému. je zřejmé, že čím více náhodných dat bude k vlastní šifrované informaci přidáno, tím bude i celá šifra odolnější proti útoku zaměřeného na analýzu šifrovaných dat. Dodatečná náhodná informace totiž pravděpodobně nebude vykazovat stejné charakteristické vlastnosti jako například šifrovaný text v angličtině zapsaný pomocí ASCII, u kterého je znám index koincidence[1], relativní výskyt jednotlivých písmen a znaků apod. Nejjednodušším řešením je střídání buňky s šifrovanou informací a buňky obsahující náhodné číslo určující další směr růstu. Z jednoho stejného čísla pak vychází obě buňky při vlastním růstu. Zde zatím nedochází k žádné redukci náhodné informace. Vlastní velikost zašifrovaného textu je dvojnásobná jak u originálu.

Aby k nějaké redukci došlo, lze střídat jednu buňku obsahující číslo s určením náhodného směru s více buňkami s vlastní nesenou informací. Poměr mezi „informačními“ a „směrovými“ buňkami je zapotřebí vhodně zvolit. Na jednu stranu je sice vhodné na jednu buňku s náhodným číslem navázat hodně buněk s informací, a to kvůli vyšší redukci, ale na druhou stranu je zapotřebí mít neustále na paměti bezpečnost šifry. Vhodným kompromisem v popisovaném případě je počet pěti informačních buněk na jednu směrovou. Jde jít ale ještě dále a nevytvářet pouze jedinou posloupnost buněk rozrůstajících se po matici. Pokud by se hned od začátku začalo vytvářet například šest samostatných posloupností buněk (tj. nejdříve podle klíče umístit nezávisle na sobě prvních šest buněk a potom při každé periodě – tj. 6 jednotlivých kroků při kterých se každá z buněk jednou rozroste – použít nový směr pro všech 6 buněk), je možné buňku obsahující informaci o dalším růstu směru rozdělit na jednotlivé bity. Nyní se v každé ze šesti buněk periody nachází pět bitů vlastní nesené informace a jeden bit směrový. Směr orientace nové periody buněk se určí ze směrových bitů buněk zpracované periody. Z důvodu větší bezpečnosti lze měnit pozici směrového bitu v buňce, jako je například znázorněno na obr. 8.

[1] Poměr počtu dvojic znaků, které jdou vytvořit k celkovému počtu dvojic znaků ve vlastním zašifrovaném textu, jinak řečeno index koincidence zašifrovaného textu udává součet relativních frekvencí dvojic znaků a aproximuje teoretickou hodnotu součtu druhých mocnin všech počtů výskytu dvojic znaků v textu.



Obrázek 8: Rozložení bitů s informací o směru růstu do buněk jedné periody

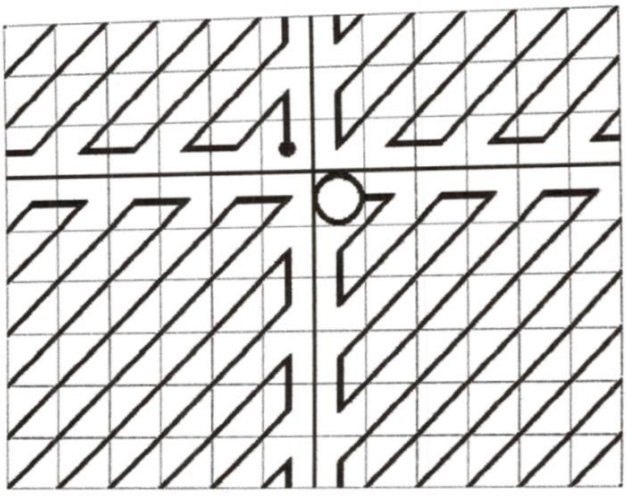
Existují ještě další metody, jako například odkázat se v hesle či v nějakém klíči na externí data, která nejsou v šifrovaném textu obsažena a ty potom k růstu buněk použít. Například lze do zašifrovaného souboru přidat index a číslo. Index potom bude odkazovat na knihu (třeba v elektronické podobě) a číslo na stránku či stať, která se použije jako vstup do generátoru pseudonáhodných čísel, jež budou postupně určovat v šifře směr růstu buněk. Tyto a další podobné metody ale vyžadují technické zázemí a spadají do kategorie málo přístupné běžnému uživateli. Navíc i ony mají svá místa a nevýhody, se kterými je zapotřebí počítat.

Transformace matice

V okamžiku, kdy je matice s konečnou platností vyplněna, nastává poslední fáze, ve které jsou data v ní obsažená převedena do formy výstupního souboru, nebo poskytnuta k dalšímu zpracování (např. pomocí počítačového rozhraní, sítě atd.) Tato, na první pohled rutinní záležitost, se ale může stát velice významnou částí celého kryptovacího algoritmu, neboť může výrazně zvýšit jeho bezpečnost. Jedná se o to, že bezpečnost kryptovacího algoritmu Night Crypt z velké části závidí na skutečnosti, že z důvodu zpětného získání zašifrované informace je zapotřebí zrekonstruovat celý buněčný růst v matici, a to od první buňky až po poslední bit. Proč tomu tak je, je popsáno v následujících kapitolách. Metody, které budou nyní uvedeny, provádějí transformaci již vyplněné matice do posloupnosti bytů (přesněji řečeno bitů) takovým způsobem, aby bez znalosti klíče bylo minimálně obtížné (v lepším případě v reálném čase nemožné) zrekonstruovat výslednou matici z dat, která jsou zašifrovaná pomocí algoritmu Night Crypt.

Realizačně nejsnadnější je načítat postupně celé řádky (popř. sloupce) matice. Případně lze tento algoritmus jemně modifikovat tak, že řádky či sloupce nejsou načítány ve vzestupném či sestupném pořadí, ale podle pravidel, které jsou závislé například na hodnotě, která je vygenerována nebo obsažena v klíči. Také lze zkombinovat načítání řádků a sloupců, takže při každém kroku se buď výška nebo šířka matice zmenší o jedna.

Složitější a účinnější je metoda cik-cak používaná například při jedné z kompresí grafických souborů BMP. V případě algoritmu Night Crypt se určí pomocí klíče bit, který se stane výchozím. Algoritmus si podle udané pozice bitu vytvoří imaginární hranici matice a bity, které seřazuje do výsledného formátu (pravděpodobně po 8 bitech = 1 Byte) načítá způsobem znázorněným na obrázku 9. Z důvodů zvýšení bezpečnosti je možné ještě navíc aplikovat při načítání bitů matice proudovou šifru.



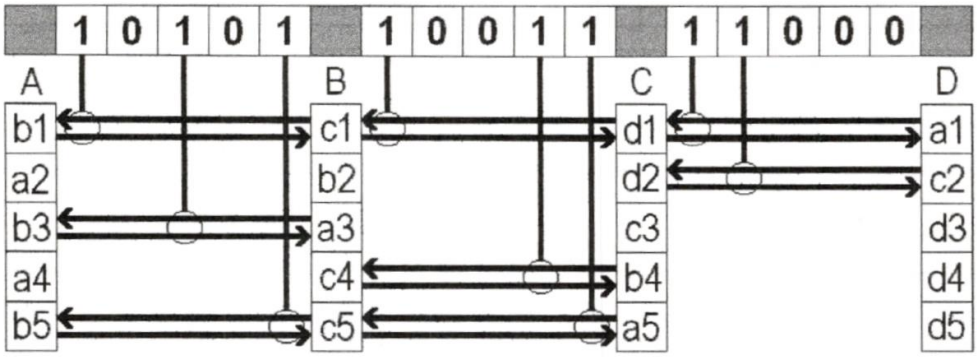
Obrázek 9: Transformace matice metodou Cik-cak

Další možností je použít před zpětnou transformací ještě jednou vlastní algoritmus Night Crypt, ale bez přidané náhodné informace. Tato metoda je ovšem výrazně pomalejší. při dešifrování je potom zapotřebí nejdříve ze zašifrovaného textu získat zpětnou transformací zaplněnou matici a v ní zrekonstruovat buněčný růst pro získání původní matice. Metod na převedení zaplněné matice bitů do výsledného formátu může být mnoho. Jejich kvalita se výrazně podílí na celkové bezpečnosti kryptovacího algoritmu Night Crypt.

Proudová šifra, heslo

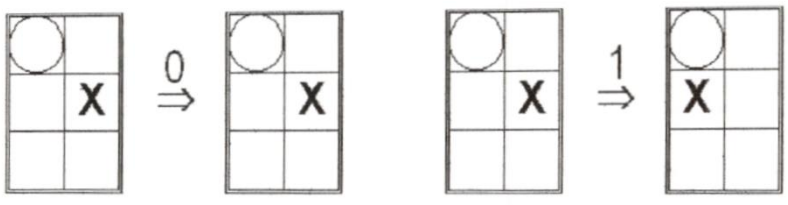
Uplatnění proudové šifry v algoritmu Night Crypt není pouze při transformaci zaplněné matice do formátu výstupních dat. Lze ji výhodně použít nejen na vstupní data, která budou již v proudově zašifrovaném tvaru uložena do informačních bitů buněk, ale také na změnu pozice směrového bitu ve vlastní buňce. K použití proudové šifry je ale zapotřebí informace, která by určovala vlastní chování proudové šifry. Informace může být získána interně přímo ze vstupních dat nebo externě, buď z použitého klíče (např. jeho rotací) anebo z hesla, které bude zadáno zvlášť. Každá z metod má své pro a proti a je na zvážení v konkrétním případě, jaký způsob je nejvhodnější použít. Vlastní implementaci lze ale navrhnout bez znalosti, jakou formou bude tato informace získána.

Jelikož Night Crypt vnímá vstupní data jako posloupnost bitů, kterou rozkládá (ve výše zmíněném příkladě) do pětic bitů, přistupuje se k heslu také jako k posloupnosti bitů. Zpracování jednoho bitu vstupní informace vyžaduje jeden bit získaný z hesla. Na zpracování jedné pětice vstupní informace je tedy zapotřebí pět bitů z hesla. Na obrázku 10 je znázorněno, jak proudová šifra pracuje.



Obrázek 10: Záměna bitů proudovou šifrou

Z obrázku je zřejmé, že metoda pracuje na principu vzájemné záměny dvou bitů, které jsou od sebe posunuty o určitou konstantní vzdálenost. Tato operace jde při šifrování provádět při na načítání vstupních dat a jejich rozkladu do posloupností bitů, kterými jsou zaplňovány bity buňky s nesenou informací. mechanismus záměny bitů nelze ale použít na ten bit v buňce, který nese údaj o směru růstu, neboť ke zpětnému získání šifrované informace je zapotřebí znát celou výslednou posloupnost. To by znemožňovalo získat informaci o směru růstu z první buňky, neboť by nebyla známa hodnota bitů, se kterými byla případně zaměněna. Hodnotu těchto bitů lze totiž získat zpětnou záměnou aplikovanou na celou posloupnost. Tu jde ovšem provést jedině se znalostí posledních bitů, které ale nelze zjistit, neboť není známa jejich pozice v matici. Proto je zapotřebí zrekonstruovat celý buněčný růst, jež se ale neobejde bez znalosti směrů růstu jednotlivých buněk.



Obrázek 11: Změna pozice směrového bitu v buňce

Na bity nesoucí informaci o směru růstu buněk je aplikována jiná metoda znázorněná na obrázku 11, která ovlivňuje pozici směrového bitu v hostitelské buňce. Je také možné místo ní použít logickou funkci **xor**. Obě metody se mohou vzájemně doplňovat, přičemž jako druhý argument k funkci **xor** lze použít paritu již spárovaných buněk. Změnu pozice směrového bitu v případě, že má dojít k záměně, vyjadřuje rovnice:

$$\text{pozice\_bitu} = ((\text{pozice\_bitu} + 2) \bmod 6) + 1 \quad (2)$$

Rozměry a velikost matice, komprese

Neboť matice neobsahuje pouze vlastní šifrovaná data, ale je vyplněna navíc bity s náhodnou hodnotou, které se využívají pro určení směru růstu buněk, velikost výstupních dat je větší jak vlastní informace před použitím algoritmu Night Crypt. O tomto negativním jevu již byla učiněna zmínka v kapitole **Nesená a přidaná informace** v souvislosti s nalezením vhodného poměru mezi objemem přidané náhodné informace a bezpečností kryptovacího algoritmu. Omezením vlivu přidání náhodných čísel k vlastním šifrovaným datům a určením poměru objemu matice a velikosti náhodné informace do ní přidané se zabývá tato kapitola.

Řešením problému je použití vhodné kompresní metody. tu je možné umístit jak před vlastní kryptovací mechanismus – komprimuje vstupní data – nebo po něm – komprimuje zašifrovaná data – anebo je také možné ji zakomponovat přímo do vlastního algoritmu. Každá ze jmenovaných variant má své přednosti a jediným jejich nedostatkem je zpomalení běhu celé aplikace.

V případě, že komprese bude aplikována již na vstupní data, je možné zvolit takovou kompresní metodu, která je nejúčinnější na předpokládaný charakter dat. Jestliže algoritmus Night Crypt bude zpracovávat pouze texty, je vhodné použít takovou metodu, která vyniká právě při zpracování textových dat. A v případě, že bude tato komprese modifikována způsobem, který eliminuje charakteristické rysy předpokládaného užitého jazyka, je navíc zvýšena bezpečnost celého algoritmu (např. tím, že při hardwarovém útoku na šifru je obtížné či nemožné zjistit, zdali je konkrétní posloupnost rozšifrovaných dat reálným textem či nesmyslem). V případě, že není předem známo, jaká data budou šifrována a tudíž i komprimována, je nutné použít takovou metodu, která není závislá na charakteru komprimovaných dat (RLE, Lempel-Ziv-Welchův algoritmus, Huffmanovo kódování apod.)

Přílohy

Seznam příloh

- Příloha 1 – všechny možné způsoby růstu nové buňky
  1.) Všechny možné způsoby růstu nové buňky s orientací 0
  2.) Všechny možné způsoby růstu nové buňky s orientací 1
  3.) Všechny možné způsoby růstu nové buňky s orientací 2
  4.) Všechny možné způsoby růstu nové buňky s orientací 3

- Příloha 2 – Růst buněk v matici
  1.) Prázdná matice s buňkou orientovanou ve směru 0
  2.) Prázdná matice s buňkou orientovanou ve směru 1
  3.) Prázdná matice s buňkou orientovanou ve směru 2
  4.) Prázdná matice s buňkou orientovanou ve směru 3

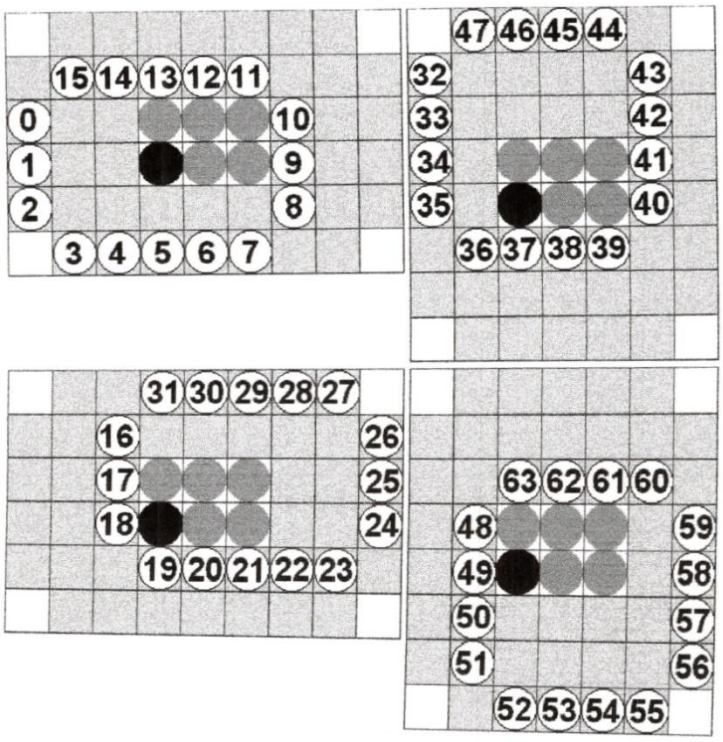
- Příloha 3 – Algoritmus vyhledávání volného místa pro novou buňku v matici
  1.) Matice s buňkami A až X
    3.1) Matice s horizontální vyhledáváním
    3.2) Matice s vertikálním vyhledáváním
  2.) Přidání buňky Y k buňce X
    3.1) Matice s horizontální vyhledáváním
    3.2) Matice s vertikálním vyhledáváním
  3.) Přidání buňky Z k buňce Y
    3.1) Matice s horizontální vyhledáváním
    3.2) Matice s vertikálním vyhledáváním
  4.) Schematické znázornění vyhledávání buňky Y a Z

Pracovní poznámky k přílohám

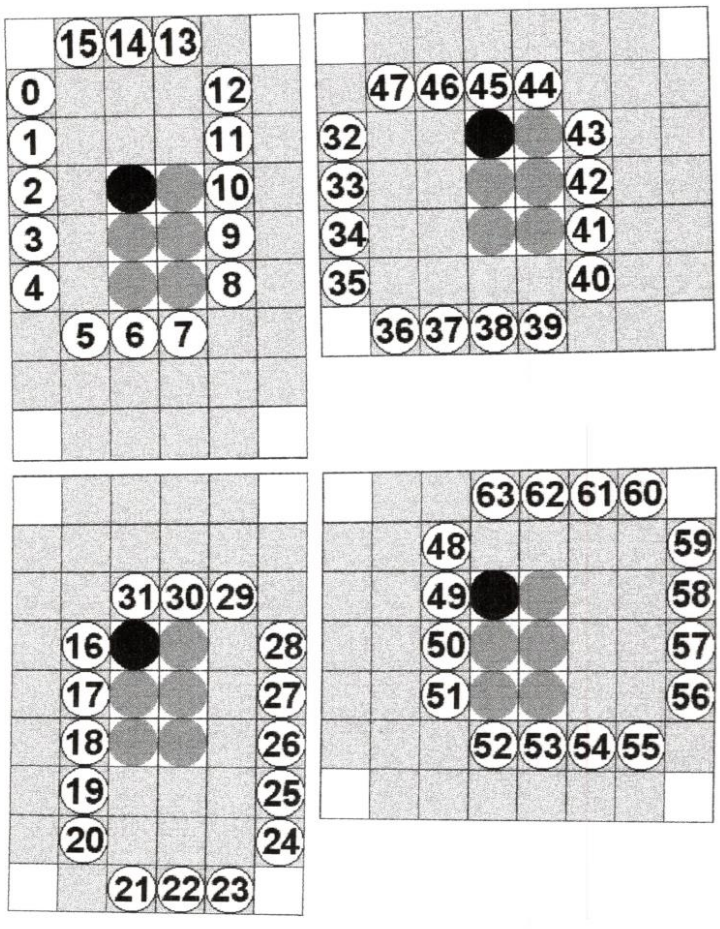
- Maximální délka matice: 225 b.
- Maximální velikost matice: 65025 b = 8129 B.
- Maximální velikost šifrovaných dat v jedné matici: 58520 b = 7315 B

Příloha 1 – Všechny možné způsoby růstu nové buňky

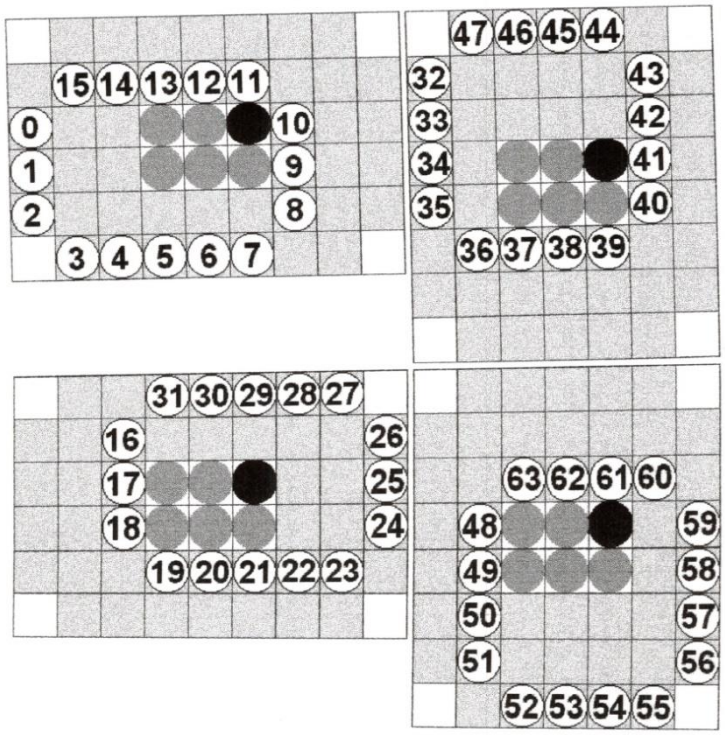
1.) Všechny možné způsoby růstu nové buňky s orientací 0



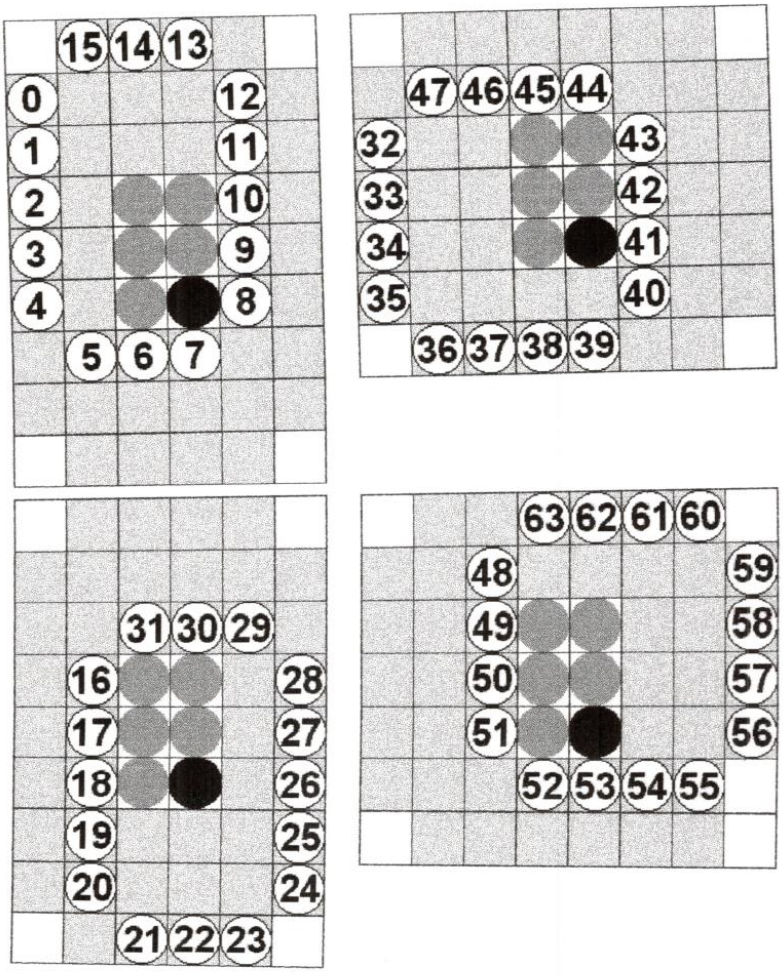
2.) Všechny možné způsoby růstu nové buňky s orientací 1



3.) Všechny možné způsoby růstu nové buňky s orientací 3

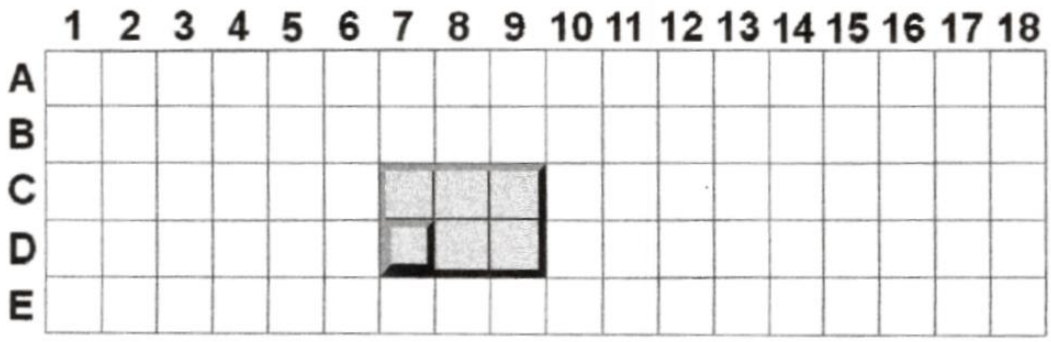


4.) Všechny možné způsoby růstu nové buňky s orientací 3

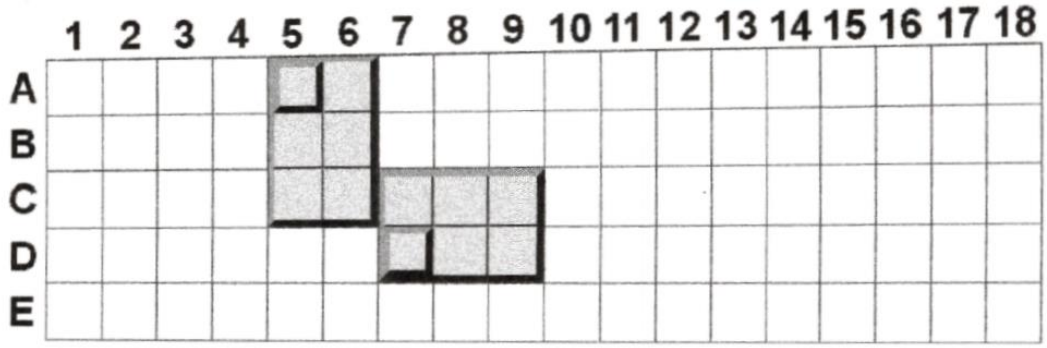


Příloha 2 – Růst buněk v matici

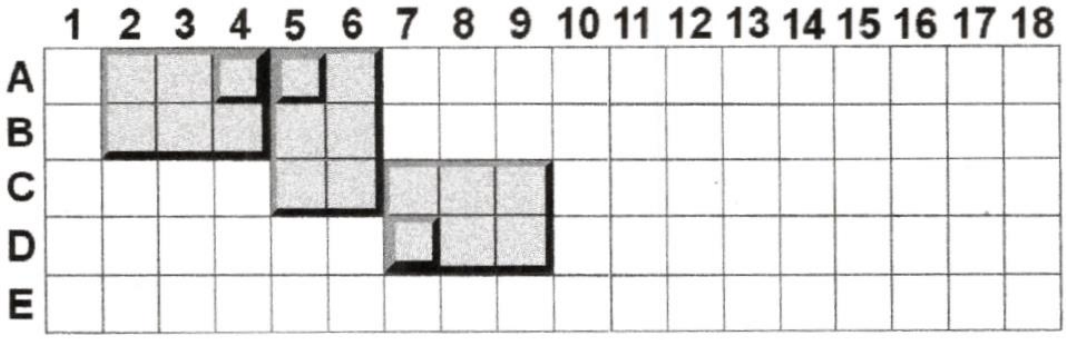
1) Prázdná matice s buňkou orientovanou ve směru 0



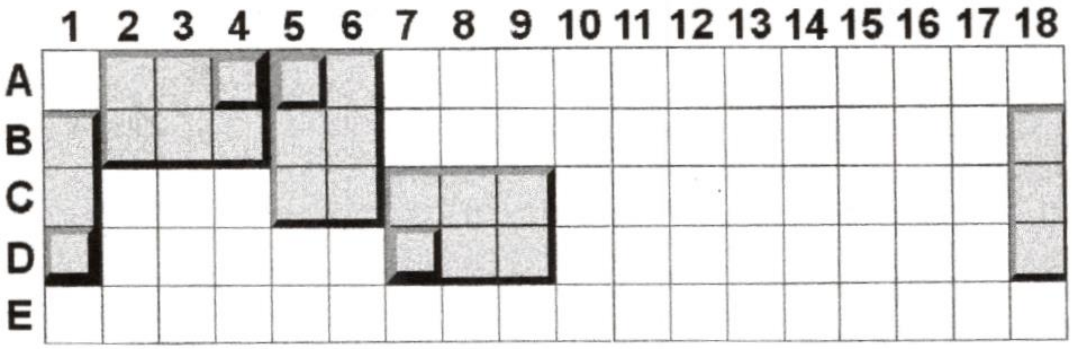
2) Prázdná matice s buňkou orientovanou ve směru 1



3) Prázdná matice s buňkou orientovanou ve směru 2

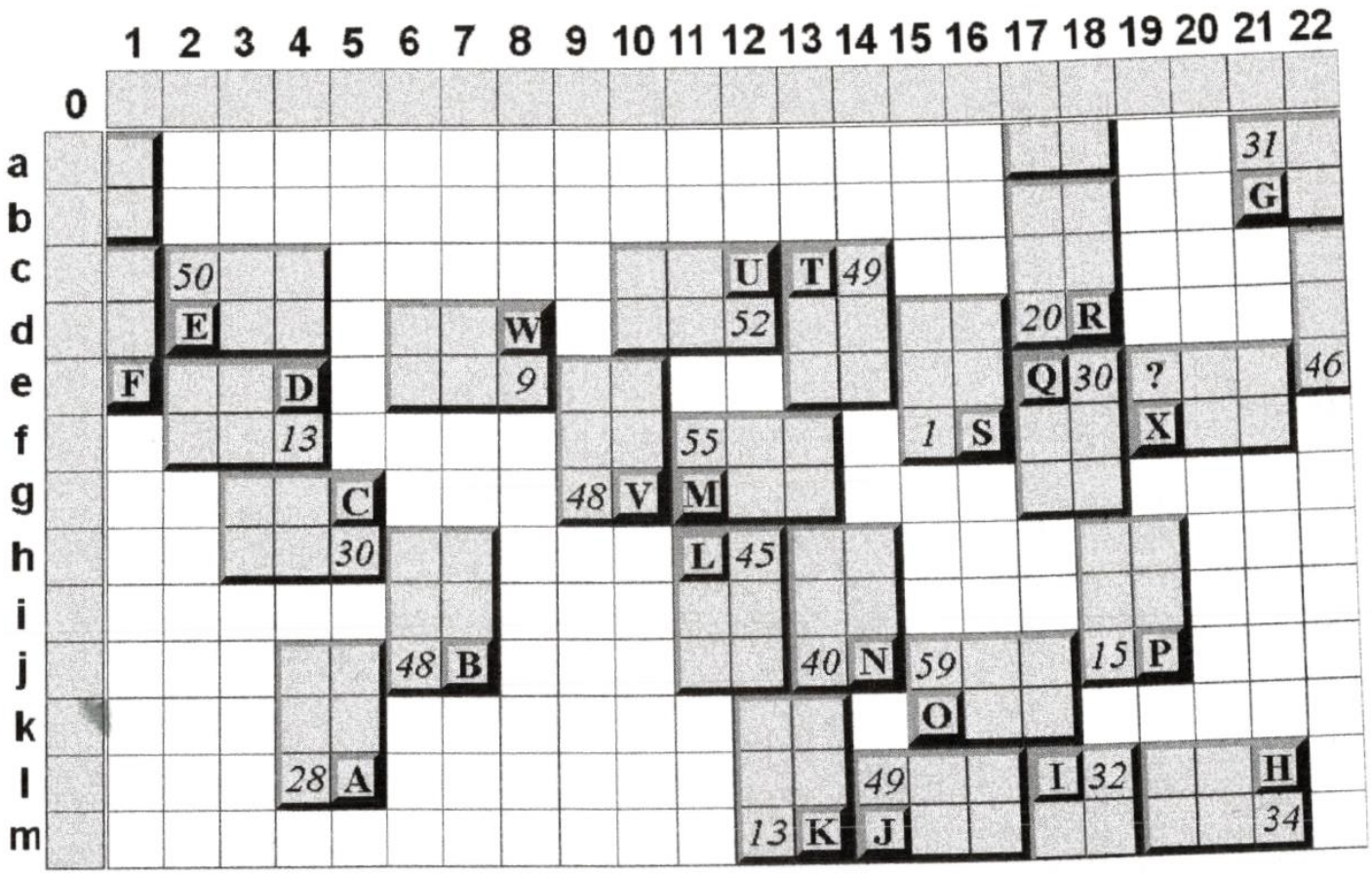


4) Prázdná matice s buňkou orientovanou ve směru 3



Příloha 3 – Algoritmus vyhledávání volného místa pro novou buňky matici

1) Matice s buňkami A až X



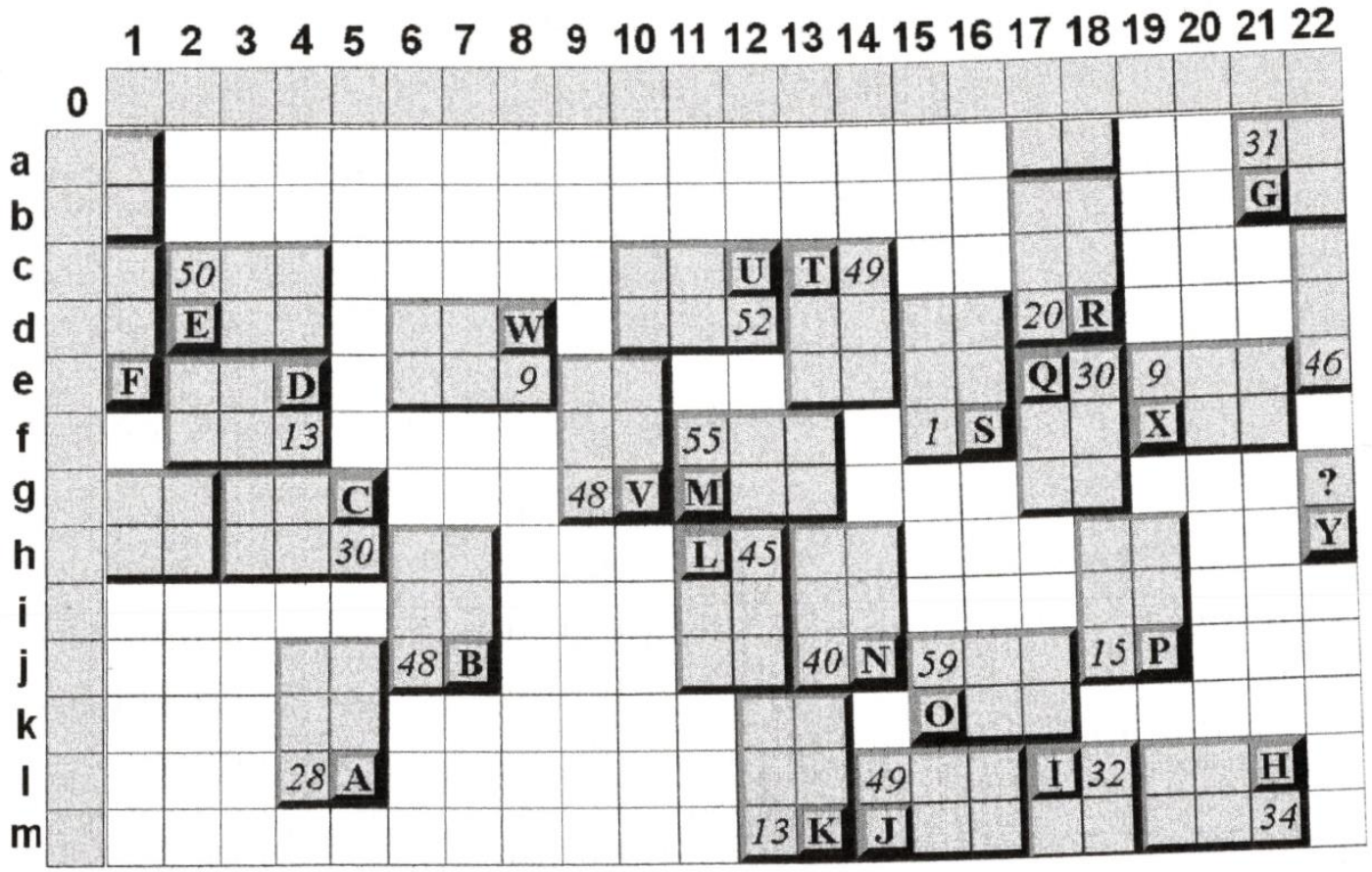
1.1) Matice s horizontální vyhledáváním

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
a	0	17																4	17			4	17
b	0	17																4	17			4	17
c	0	22	22	22	22						21	10	10	10	10			21	10				7
d	0	22	22	22	22		21	22	22		21	22	22	22	22	22	22	22	22				21
e	5	5	5	5	5		5	5	5	5	5			5	5	5	5	5	5	5	5	5	5
f	0		7	9	9					7	9	9	9	9		7	9	9	9	9	9	9	
g	0			8	3	3				16	9	9	9	9				21	17				
h	0			10	3	3	3	3				17	11	11	11				22	18			
i	0						10	6				17	11	11	11				22	18			
j	0				10	4	4	4				22	11	11	11	11	11	11	11	11			
k	0				8	4							2	12		2	12	12					
l	0				8	4							2	12	12	12	12	12	12	12	12	12	
m	0												2	12	12	12	12	12	12	12	12	12	

1.2) Matice pro vertikální vyhledáváním

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
a		h																q	r			l	h
b		a																q	r			l	a
c		a	i	k	b						j	m	l	m	n			q	r				a
d		a	c	c	c		m	m	h		c	c	l	m	n	i	i	q	r				a
e		a	c	c	c		d	d	d	j	c			m	n	d	d	q	r	c	i	i	a
f			c	c	c					e	c	m	l	m		d	d	q	r	e	e	l	
g				c	c	b				e	c	c	l	m				q	r				
h				c	c	g	m	m				c	l	m	n				r	c			
i							d	d				c	l	m	n				r	e			
j					b	b	d	d				c	l	m	n	c	c	q	r	e			
k					c	g							l	m		j	j	q					
l					c	g							l	m	n	j	j	q	r	c	c	i	
m													l	m	n	j	j	q	r	e	l	l	

Přidání buňky Y k buňce X



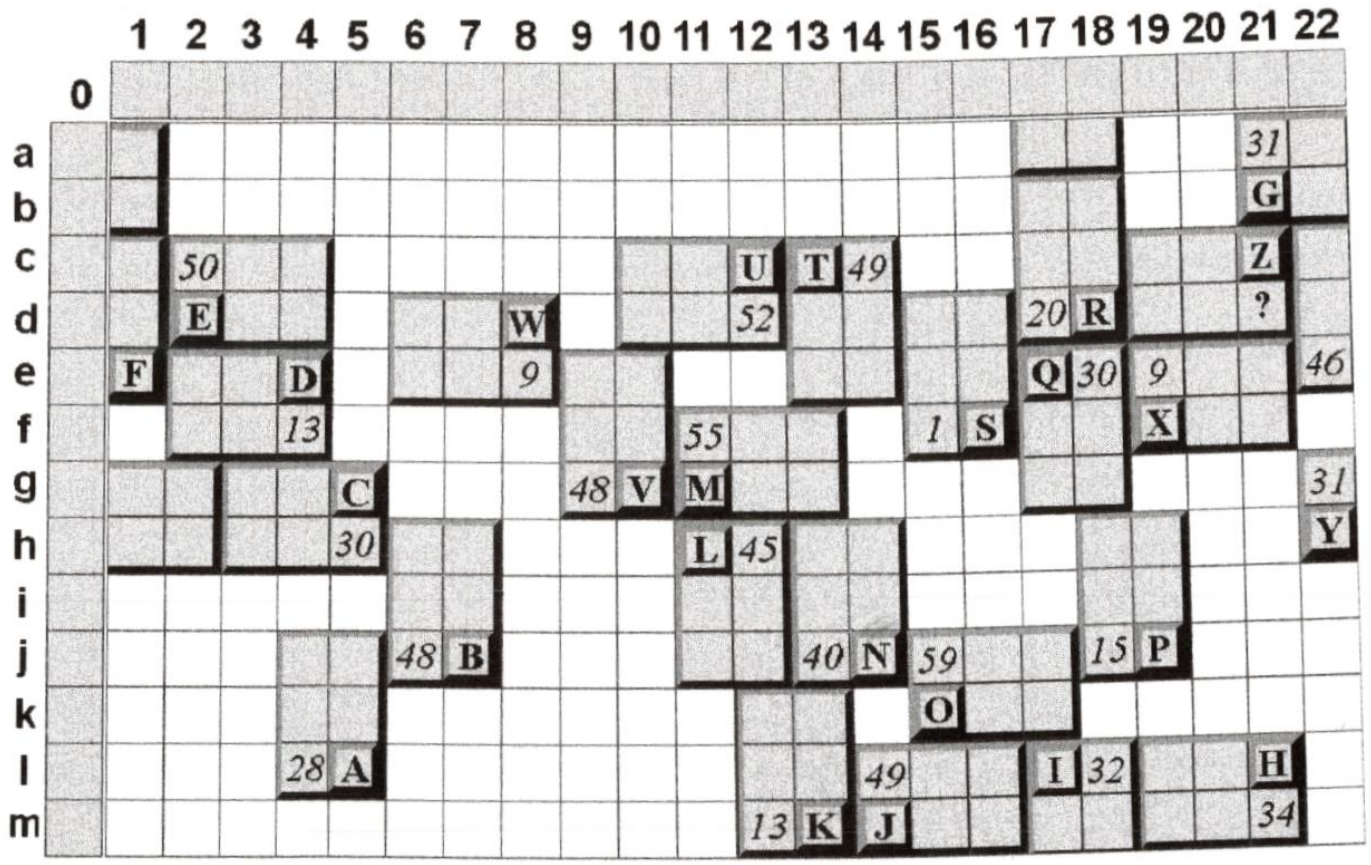
2.1) Matice s horizontálním vyhledáváním po přidání buňky Y

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
a	0	17																4	17			4	17
b	0	17																4	17			4	17
c	0	22	22	22	22						21	10	10	10	10			21	10				7
d	0	22	22	22	22		21	22	22		21	22	22	22	22	22	22	22	22				21
e	5	5	5	5	5		5	5	5	5	5			5	5	5	5	5	5	5	5	5	5
f	0		7	9	9					7	9	9	9	9		7	9	9	9	9	9	9	
g	0	22	22	22	22	22				16	9	9	9	9				21	17				8
h	0	18	18	18	18	18	18	18				17	11	11	11				10	18			10
i	0						10	6				17	11	11	11				22	18			
j	0				10	4	4	4				22	11	11	11	11	11	11	11	11			
k	0				8	4							2	12		2	12	12					
l	0				8	4							2	12	12	12	12	12	12	12	12	12	
m	0												2	12	12	12	12	12	12	12	12	12	

2.2) Matice s vertikálním vyhledáváním po přidání buňky Y

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
	0	0	0	0	0	0	0	0	0	0	0	0	l	m	n	0	0	q	r	0	0	0	0
a		k																q	r			l	k
b		a																q	r			l	a
c		a	k	k	b						j	m	l	m	n			q	r				a
d		a	c	c	c		m	m	h		c	c	l	m	n	i	i	q	r				a
e		a	c	c	c		d	d	d	j	c			m	n	d	d	q	r	c	i	i	a
f			c	c	c					e	c	m	l	m		d	d	q	r	e	e	l	
g		k	c	c	c	b				e	c	c	l	m				q	r				k
h		a	c	c	c	g	m	m				c	l	m	n				r	c			a
i							d	d				c	l	m	n				r	e			
j					b	b	d	d				c	l	m	n	c	c	q	r	e			
k					c	g							l	m		j	j	q					
l					c	g							l	m	n	j	j	q	r	c	c	i	
m													l	m	n	j	j	q	r	e	l	l	

3) Přidání buňky Z k buňce Y



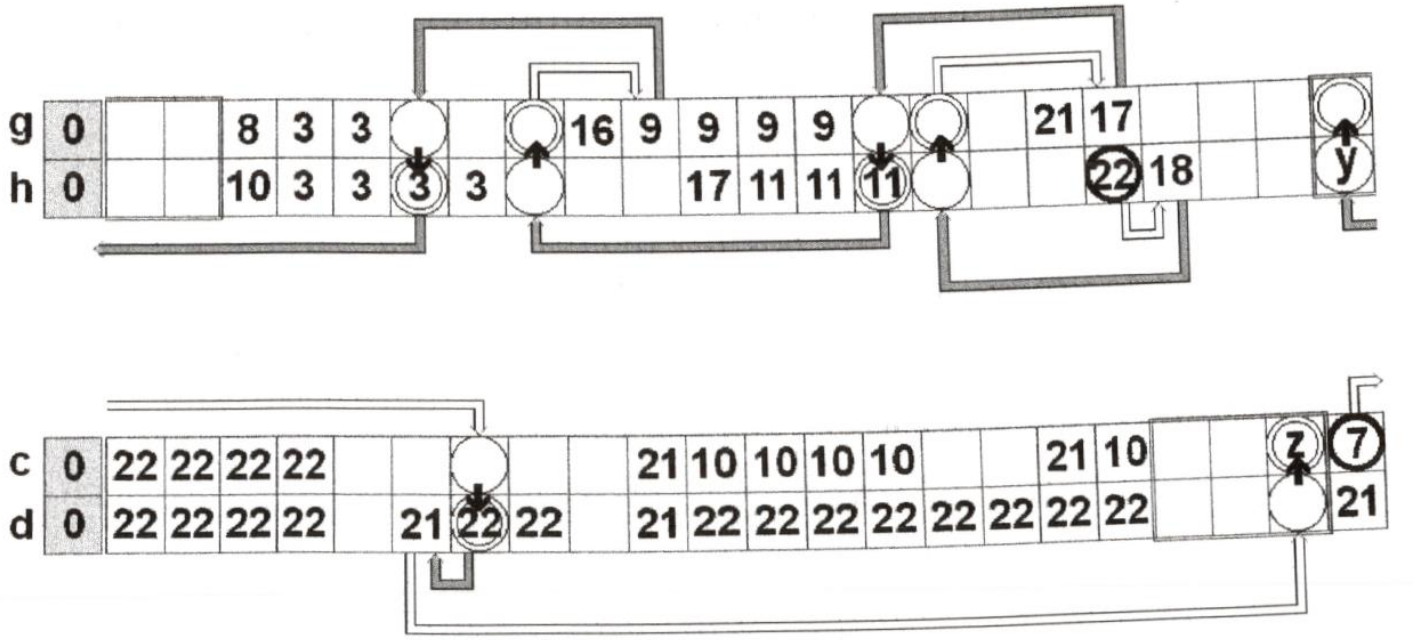
3.1) Matice s horizontálním vyhledáváním po přidání buňky Z

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
	0																						
a	0	17																4	17			4	17
b	0	17																4	17			4	17
c	0	10	10	10	10						7	10	10	10	10			7	10	10	10	10	10
d	4	4	4	4	4		4	4	4		4	4	4	4	4	4	4	4	4	4	4	4	4
e	5	5	5	5	5		5	5	5	5	5			5	5	5	5	5	5	5	5	5	5
f	0		7	9	9					7	9	9	9	9		7	9	9	9	9	9	9	
g	0	22	22	22	22	22				16	9	9	9	9				21	17				8
h	0	18	18	18	18	18	18	18				17	11	11	11				10	18			10
i	0						10	6				17	11	11	11				22	18			
j	0				10	4	4	4				22	11	11	11	11	11	11	11	11			
k	0				8	4							2	12		2	12	12					
l	0				8	4							2	12	12	12	12	12	12	12	12	12	
m	0												2	12	12	12	12	12	12	12	12	12	

3.2) Matice s vertikálním vyhledáváním po přidání buňky Z

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
	0	0	0	0	0	0	0	0	0	0	0	0	l	m	n	0	0	q	r	s	0	0	0
a		k																q	r			l	k
b		a																q	r			l	a
c		a	k	k	b						j	m	l	m	n			q	r	s	i	l	a
d		a	c	c	c		m	m	h		c	c	l	m	n	i	i	q	r	s	l	l	a
e		a	c	c	c		d	d	d	j	c			m	n	d	d	q	r	s	l	l	a
f			c	c	c					e	c	m	l	m		d	d	q	r	s	l	l	
g		k	c	c	c	b				e	c	c	l	m				q	r				k
h		a	c	c	c	g	m	m				c	l	m	n				r	s			a
i							d	d				c	l	m	n				r	s			
j					b	b	d	d				c	l	m	n	c	c	q	r	s			
k					c	g							l	m		j	j	q					
l					c	g							l	m	n	j	j	q	r	s	i	i	
m													l	m	n	j	j	q	r	s	l	l	

4) Schematické znázornění vyhledávání buňky Y a Z



Jiří Altior
Night Crypt: šifrovací algoritmus

Výňatek z diplomové práce[2]
Jiřího Vyšína (dnes Altior)
na SPŠE a VTŠ Pardubice
(dnes SPŠE a VOŠ Pardubice,
Karla IV. 13, 530 02 Pardubice I)
1999

Text: původní znění, přepsáno
Korektury: opraveny gramatické překlepy
Ilustrace: vloženy dochované původní ilustrace[3]

INSTITUT RENESANCE KULTURNÍHO DĚDICTVÍ z. ú.
Publikační sekce a sekce aplikované metafyziky
1. PDF vydání
20 stran
Praha, **MMXXII** ©

[2] Diplomová práce byla obhájena s hodnocením 1 a škola ukončena s vyznamenáním.
[3] Kopie diplomové práce, jež byla v držení autora práce, byla zcizena. Dochovala se pouze její část, která je zde uvedena.