Masarykova univerzita v Brně
Fakulta informatiky

Problematika náhodných a pseudonáhodných sekvencí v kryptografických eskalačních protokolech a implementacích na čipových kartách

Diplomová práce

Jan Krhovják

16. května 2005

Prohlašuji, že tato práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

V Brně dne 16. května 2005 .....

Mé poděkování patří Dr. Václavu Matyášovi, jehož odborné vedení a cenné rady významně přispěly při vzniku této práce.

Shrnutí

V teoretické části práce si popíšeme principy několika základních kryptografických eskalačních protokolů a seznámíme se s problematikou generování náhodných a pseudonáhodných sekvencí v implementacích na čipových kartách. Praktickou částí práce bude test několika generátorů dostupných na vybraných čipových kartách.

Klíčová slova

Eskalační protokoly, PRBG, PRNG, RBG, RNG, testy náhodnosti, TRNG.

Obsah

Kapitola 1

Úvod

V dnešní době, kdy se počítače stávají běžnou součástí každodenního života a elektronické obchodování již není jen předmětem teoretických rozprav několika nadšenců v akademické sféře, je vzdálená autentizace uživatelů naprosto klíčovým faktorem mnoha používaných systémů a potřeba kvalitních a bezpečných autentizačních protokolů vzrůstá každým dnem stále více.

K velmi oblíbené metodě autentizace uživatelů patří použití nějaké tajné, krátké, a tudíž i snadno zapamatovatelné informace, jako je například PIN či heslo, jejíž znalostí prokazuje uživatel svou identitu. Naneštěstí klasické kryptografické protokoly navržené pro vzdálenou autentizaci vyžadují použití kvalitních a předem ustavených kryptografických klíčů, které bývají dlouhé řádově stovky bitů a pro člověka jsou zcela nezapamatovatelné.

Využití krátkých PINů či hesel v těchto protokolech bylo dlouhou dobu pokládáno za nemožné, protože jejich verifikace přes nezabezpečený kanál je vždy vystavovala riziku slovníkového útoku. Počátkem devadesátých let minulého století však bylo objeveno několik metod, jak se tohoto útoku vyvarovat. Vznikla nová třída autentizačních tzv. eskalačních protokolů, které byly založeny pouze na použití PINů či hesel a umožňovaly navíc i ustavení kvalitních kryptografických klíčů.

Protože po uživateli je celkově vyžadována jen znalost nějaké tajné informace, jsou tyto protokoly snazší a méně náročné při zavádění do praxe, něž například metody založené na použití biometrik či tokenů. Tam kde již nějaká forma bezpečnostní infrastruktury existuje, může dodatečná implementace těchto protokolů sloužit jako další nezávislý autentizační mechanizmus, který významně přispěje a posílí zabezpečení systému jakožto celku.

Praktická aplikace tohoto typu protokolů pokrývá veškeré případy komunikace přes nezabezpečený kanál, kde by dlouhotrvající uchovávání kryptografických klíčů bylo nebezpečné či nepraktické. Příkladem může být jejich nasazení v dnes poměrně rozšířeném autentizačním sytému Kerberos, ale také použití jako náhrada zastaralých internetových protokolů umožňujících vzdálený login pomocí hesel zasílaných v otevřené podobě. Zajímavou aplikací může být i zabezpečení vzdáleného přístupu ke kryptografické čipové kartě podporující technologii Java Card™ či vytvoření efektivnějšího modelu budoucích sítí bankomatů, kde by již autentizace pomocí zákazníkova PINu neprobíhala přes několik mezilehlých síťových přepínačů sdílejících stejné předem ustavené tajné šifrovací klíče.

Pokud se jedná o kvalitu výsledného kryptografického klíče, tak nezávisí jen na jeho bitové délce (ačkoliv ta o něm mnohé vypovídá) či na návrhu protokolu samotného. Základním požadavkem je totiž náhodnost, a ta závisí především na generátorech náhodných či pseudonáhodných sekvencí, které jednotlivé strany v průběhu protokolu používají. Klíč může vzniknout na základě náhodných dat jedné či dokonce více stran, a bezpečnost (tj. nepředvídatelnost) použitých generátorů je proto naprosto zásadní.

Získávání skutečně náhodných sekvencí je typicky založeno na nedeterministických fyzikálních jevech (jako například rozpad radioaktivních látek) a jejich generování je v deterministickém prostředí počítačových systémů extrémně obtížné a pomalé (tj. v rozumném čase lze vygenerovat pouze velmi krátké skutečně náhodné sekvence). Proto se velmi často využívají pseudonáhodné sekvence, které jsou deterministicky generovány z kratších skutečně náhodných vstupů. Generátory pseudonáhodných sekvencí jsou obvykle řádově rychlejší a jimi vygenerované sekvence mají také lepší statistické vlastnosti. Protože jsou však tyto generátory vždy deterministické, je veškerá náhodnost výsledných sekvencí redukována na náhodnost vstupu. Existuje velká spousta generátorů pseudonáhodných čísel (např. pro účely simulací, testování apod.), ale oproti nim by generátory určené pro kryptografické účely měly produkovat pseudonáhodné sekvence výpočetně nerozlišitelné od skutečně náhodných sekvencí.

Na první pohled by se mohlo zdát, že eskalační protokoly mají s generátory náhodných či pseudonáhodných sekvencí jen pramálo společného. Jistá paralela zde ale přeci jen existuje. U eskalačních protokolů si uživatel volí krátký a snadno zapamatovatelný PIN či heslo, na jehož základě je pak ustaven kvalitní (a tedy i dostatečně dlouhý) kryptografický klíč. Podobně generátor skutečně náhodných sekvencí produkuje v rozumném čase pouze krátké skutečně náhodné sekvence, na jejichž základě se pak vytvářejí kvalitní (a dostatečně dlouhé) pseudonáhodné sekvence. Rozdíl je pouze v tom, že v případě eskalačních protokolů hraje zásadní roli především náhodnost a nepředvídatelnost výsledného klíče (a až poté jeho délka), kdežto u generátorů pseudonáhodných čísel jde především o efektivní vytvoření nepředvídatelné dostatečně dlouhé pseudonáhodné sekvence (a její náhodnost závisí pouze na náhodnosti vstupu).

Rozvržení kapitol

V druhé kapitole se blíže seznámíme s principy několika základních eskalačních protokolů a s jejich bezpečnostními nedostatky. V třetí kapitole se pak budeme věnovat problematice generování (pseudo)náhodných sekvencí se zaměřením na implementace v čipových kartách. V kapitole čtvrté nakonec popíšeme průběh testování kvality několika generátorů dostupných na vybraných čipových kartách.

Tato práce se opírá o základní znalosti v oblasti algebry [Ros00], aplikované kryptografie [MvOV01] a statistiky [BMO01].

Kapitola 2

Eskalační protokoly

Zajímavý způsob vytváření (a distribuce) kvalitních kryptografických klíčů nám nabízejí některé kryptografické protokoly. Jejich typickým příkladem je známá Diffie-Hellmanova (DH) metoda ustavení klíčů [DH76], kde sdílený klíč vzniká na základě náhodných dat vygenerovaných oběma stranami.

Chtějí-li si dvě strany[1] protokolu ustavit sdílený klíč, domluví se nejprve nějakým způsobem na velkém prvočíselném modulu β a na generátoru α multiplikativní grupy $\mathbb{Z}_\beta^*$. Obě tyto hodnoty jsou veřejné[2] a jako β je doporučeno volit *bezpečné prvočíslo* tvaru $\beta = 2\gamma+1$, kde γ je velké prvočíslo [PH78, vOW96]. Poté si přes nezabezpečený kanál zašlou následující zprávy (někdy označované jako DH hodnoty):

$$\begin{array}{lcl} \boxed{\text{A}} & & \boxed{\text{B}} \\ \alpha^{r_A} \mod \beta & \longrightarrow & \\ & \longleftarrow & \alpha^{r_B} \mod \beta \\ & \dots & \end{array}$$

Čísla r_A i r_B jsou jednotlivými stranami generovány náhodně[3] z intervalu $\langle 2, \beta-1\rangle$. Sdílený klíč sezení K_S je pak každou stranou získán tak, že je přijatá DH hodnota umocněna na náhodně vygenerované číslo příslušné strany a $K_S = \alpha^{r_A r_B} \mod \beta$.

Nedostatkem tohoto protokolu je zranitelnost útokem typu *man in the middle*. Tento problém řeší autentizované verze protokolu [DvOW92], které umožňují vzájemné ověření identity jednotlivých komunikujících stran.

My se v této kapitole zaměříme především na tzv. *eskalační protokoly*, které jsou založeny na použití dat s nízkou entropií (jako například hesel). Tyto kryptografické protokoly náleží k mimořádné třídě metod, které zajišťují autentizované ustavení klíčů přes nezabezpečený kanál a jsou založeny na použití hesel způsobem, který je nevystaví riziku off-line útoku hrubou silou (a tedy ani slovníkovému útoku). Jako eskalační je nazýváme proto, že málo kvalitní hesla eskalují na kvalitní kryptografické klíče. Pokud neřekneme jinak, budeme předpokládat, že obě strany protokolu (tj. klient i server) mají ustaveno společné tajné heslo. Při vlastním popisu protokolů budeme také velmi často vynechávat zasílané identifikátory jednotlivých stan.

[1] Většinou je budeme označovat pojmy *strana A* a *strana B*. Při použití terminologie klient/server, pak bude strana A chápána vždy jako klient a strana B jako server.

[2] Při popisu tohoto či jemu podobných protokolů budeme předpokládat, že již byly ustaveny.

[3] Předpokládáme tedy na obou stranách existenci kryptograficky bezpečného generátoru.

2.1 (DH)EKE

Historicky první protokol spadající do této kategorie je označován jako *EKE (encrypted key exchange)* [BM92] a jedná se o zcela originální kombinaci asymetrické a symetrické kryptografie. Základní část protokolu vypadá následovně:

$$\begin{array}{cccc} \boxed{\text{A}} & & & \boxed{\text{B}} \\ E_P(V_A) & \longrightarrow & & \\ & \longleftarrow & & E_P(E_{V_A}(K_S)) \\ & \ldots & & \end{array}$$

Sdílený tajný klíč (v tomto případě tedy heslo P) je použit k zašifrování veřejného klíče strany A (z jednorázově vygenerovaného páru soukromý/veřejný klíč), který je v zašifrované podobě doručen straně B, dešifrován a použit spolu s heslem k zašifrování touto stranou jednorázově vygenerovaného klíče sezení. Ten je pak odeslán zpět straně A a pomocí hesla a soukromého klíče dešifrován. V další části protokol pokračuje výměnou několika klíčem sezení zašifrovaných zpráv, které zajišťují ochranu proti útokům přehráním (např. pomocí náhodných čísel či časových razítek) a ověřují, zda ustavení klíče sezení proběhlo korektně.

EKE může být použit jak se systémy umožňujícími distribuci veřejného klíče (obzvláště dobře funguje s DH metodou ustavení klíčů [DH76]) tak s asymetrickými kryptosystémy (po vyřešení specifických problémů lze použít například RSA či ElGamal). Z bezpečnostního hlediska je u tohoto protokolu zcela zásadní, aby zpráva, která má být heslem zašifrována (např. výše zmíněný veřejný klíč), byla nerozlišitelná od náhodného čísla. V opačném případě (tj. kdyby zpráva měla určitou strukturu, kontrolní součet apod.) by bylo snadné provést off-line útok hrubou silou.

Při implementaci EKE pomocí RSA, není například možné efektivně zakódovat veřejný klíč (n,e) tak, aby byl nerozlišitelný od náhodného čísla – útočník může vždy testovat, zdali má modulo n malé prvočíselné dělitele. Proto může být heslem zašifrován pouze exponent e, ke kterému je ještě před zašifrováním s pravděpodobností 1/2 přičtena hodnota 1 (protože všechny přípustné hodnoty e jsou liché). Implementace pomocí kryptosystému ElGamal tímto nedostatkem netrpí, protože veřejné klíče jsou zde generovány jako $\alpha^r \mod \beta$ a mají tedy rovnoměrné rozložení na intervalu hodnot $\langle 1, \beta - 1\rangle$.

Kdybychom u výše uvedeného příkladu s RSA ponechali před zašifrováním exponenty vždy liché, mohl by útočník při každém pokusu o dešifrování $E_P(V_A)$ pomocí zkoušených hesel P' zredukovat prostor možných hesel přibližně na polovinu. Prostě by jednoduše vyřadil všechna taková hesla P', která by po provedení operace $D_{P'}(E_P(V_A))$ vrátila jako výsledek sudé číslo. Takováto redukce prostoru hesel na polovinu zdánlivě nevypadá nijak nebezpečně. Uvážíme-li však, že při dalších sezeních jsou vždy vygenerovány nové hodnoty klíčů, lze tímto způsobem dále pokračovat v dělení/redukci prostoru hesel (celkově se tedy bude snižovat logaritmicky). Obecně se tento typ útoku, při němž dochází k postupnému dělení/redukci prostoru klíčů (hesel), nazývá *partition attack*.

Některé kryptosystémy jako například ElGamal mohou ale minimální dělení připustit – zde je pomocí hesla šifrováno nějaké celé číslo z intervalu $\langle 0, \beta - 1\rangle$. Pokud jej zakódujeme do n bitů, tak budeme při zkoušení hesel schopni vyřadit všechna

taková hesla P', která po provedení operace $D_{P'}(E_P(V_A))$ vrátí jako výsledek číslo z intervalu $\langle\beta, 2^n - 1\rangle$. Zřejmě pokud se β bude blížit $2^n - 1$ tak bude možno vyloučit jen malý počet hesel. Naopak, hodnoty β blízké 2^{n-1} povedou k velké redukci prostoru hesel.

Dále je také potřeba zvážit doplnění šifrovaných dat na odpovídající délku bloku podporovanou příslušným symetrickým algoritmem. Doplnění nulami s sebou přináší opět riziko útoku, a proto by tyto doplňující bity měly být raději náhodné.

Poslední dva výše zmíněné problémy lze efektivně vyřešit následující úpravou. Předpokládejme, že šifrujeme nějaké číslo modulo β a délka šifrovacího bloku je m bitů, kde $2^m > \beta$. Dále nechť $x = \lfloor 2^m/\beta \rfloor$ označuje číslo, kolikrát se interval $\langle 0, \beta-1\rangle$ vejde do jednoho bloku. Pak vždy zvolíme náhodnou hodnotu $j \in \langle 0, x-1\rangle$ a pomocí nemodulární aritmetiky přičteme k původnímu vstupnímu číslu hodnotu $j\beta$. Pokud je hodnota vstupního čísla menší než $2^m - x\beta$, použijeme $j \in \langle 0, x\rangle$.

Implementace základní části EKE pomocí DH metody ustavení klíčů se pak od obecného popisu protokolu mírně odklání – v prvních dvou krocích jsou přenášené hodnoty chráněny heslem a na jejich základě si pak nezávisle strana A i B vygeneruje klíč sezení (např. výběrem určitých bitů z $\alpha^{r_A r_B} \mod \beta$):

A		B
$E_P(\alpha^{r_A} \mod \beta)$	$\longrightarrow$	
	$\longleftarrow$	$E_P(\alpha^{r_B} \mod \beta)$
	$\dots$	

Útočník bez znalosti hesla nemůže provést útok *man in the middle* ani výpočet diskrétního logaritmu (v případě malých hodnot β). Tuto implementaci EKE také budeme někdy označovat jako DHEKE.

V [BM92] je dále diskutována volba vhodných (a)symetrických kryptosystémů, správných parametrů α, β a obrana proti kryptoanalytickým útokům.

2.2 AEKE

Nevýhodou protokolu EKE je, že jednotlivé strany si musí uchovávat svá sdílená hesla uložená v otevřené podobě. Protokol *AEKE (augmented encrypted key exchange)* [BM93] je takovým rozšířením a vylepšením protokolu DHEKE, které zajišťuje, že si server již uchovává hesla pouze jednocestně zašifrována – někdy je budeme označovat jako *verifikační hodnoty*. Útočník, který by získal přístup k souboru takto zašifrovaných hesel by sice stále mohl vystupovat jako falešný server, ale nemohl by se jejich přímým použitím vydávat serveru za libovolného uživatele (nejprve by musel provést slovníkový útok).

Implementace AEKE pomocí schémat digitálních podpisů využívá páru soukromý/veřejný klíč, který je na základě hesla jednocestně vygenerován stranou A. Veřejný klíč je jakožto verifikační hodnota sdílen také se stranou B. Protokol začíná ustavením klíče sezení K_S metodou DHEKE, kde je k symetrickému šifrování namísto hesla použit výše zmíněný veřejný klíč. Strana A poté navíc podepíše K_S svým soukromým klíčem, podpis pomocí K_S symetricky zašifruje a zašle straně B. Ta podpis dešifruje a nakonec sdíleným veřejným klíčem ověří jeho korektnost.

2.3 Interlock protocol

(A)EKE je vhodnou náhradou za Rivestův a Shamirův *interlock protocol* [RS84]. Ten je navržen tak, aby na komunikačním spoji detekoval aktivní útočníky. Davies a Price v [DP89] navrhli způsob jeho použití také k autentizaci, ale později byl na něj v [BM94] popsán útok. Z [BM94] v této části vycházíme.

Pro použití tohoto protokolu k autentizaci předpokládejme, že strany A a B sdílí dvě tajná hesla P_A a P_B. První fází protokolu je ustavení klíče K_{AB} mezi těmito stranami, čehož je dosaženo standardní DH metodou. V druhé fázi protokolu pak pomocí tohoto klíče každá strana zašle své zašifrované heslo straně druhé, která jej porovná s uloženým heslem. Aby se zabránilo útoku *man in the middle*, rozdělila se zašifrovaná hesla na dvě části, které si měly jednotlivé strany střídavě vyměnit. Popis druhé fáze protokolu je uveden níže – v hranatých závorkách je vyznačeno číslo příslušné části zašifrovaného hesla:

A		B
	$\dots$	
$E_{K_{AB}}(P_A)\langle 1\rangle$	$\longrightarrow$	
	$\longleftarrow$	$E_{K_{AB}}(P_B)\langle 1\rangle$
$E_{K_{AB}}(P_A)\langle 2\rangle$	$\longrightarrow$	
	$\longleftarrow$	$E_{K_{AB}}(P_B)\langle 2\rangle$

Pokud by nějaký útočník vstoupil do procesu ustavení klíče[4] a následné autentizace, tak by nemohl dešifrovat první polovinu zprávy přijatou od strany A dokud nedorazí i její druhá polovina – dešifrováním jedné poloviny zprávy by nezískal správné heslo. To by mu mělo definitivně zabránit zfalšovat první zprávu určenou pro stranu B, čímž se mělo útoku *man in the middle* zabránit.

Předpokládejme však, že má útočník plnou kontrolu nad komunikačními spoji mezi A a B, a že se pokouší vydávat za stranu A. Nejprve si v první fázi protokolu se stranou A ustaví společný šifrovací klíč. V druhé fázi může straně A zaslat libovolné zašifrované heslo a počkat, až mu strana A pošle druhou polovinu svého zašifrovaného hesla. Po jeho dešifrování pak přeruší komunikaci s A (případně znemožní komunikaci mezi A a B) a sám začne komunikaci se stranou B.

Tomuto útoku nezabrání ani případné modifikace protokolu tak, že by druhou fázi dialogu začínala strana B.

2.4 MEKE

V [STW95] je popsána efektivnější varianta protokolu DHEKE, často označovaná jako *MEKE (minimal encrypted key exchange)*. Optimalizací došlo k redukci počtu zasílaných zpráv i prováděných kryptografických operací. Kromě popisu MEKE jsou diskutovány také kryptoanalytické útoky na (A)EKE a obrana proti nim.

Jako podstatné je u (A)EKE zdůrazněno především bezpečné ustavení klíče sezení K_S, tak aby jeho pozdější kompromitace neumožnila slovníkový útok na heslo.

[4] Při použití samotné DH metody ustavení klíčů je útok *man in the middle* vždy možný.

Proto by měl být klíč sezení raději vypočítán z původního K'_S pomocí kryptografické hašovací funkce jako $K_S = h(K'_S)$. Protokoly DHEKE i MEKE jsou proti podobnému typu útoku odolné.

2.5 DWEKE

I přes svůj precizní návrh jsou mnohé používané protokoly stále náchylné k útokům, které umožňují se znalostí současně používaného hesla získat všechna v budoucnu ustavená hesla – tzv. *password chaining attacks*. Zásadní problém většiny těchto protokolů totiž je, že používají své heslo také k ochraně zpráv, které jsou použity k ustavení nového hesla. Útočník, který zná původní heslo, tak může s jeho pomocí snadno dešifrovat zprávu obsahující nové heslo. Výsledkem odhalení byť jen jediného hesla je pak kompromitace veškeré komunikace daného uživatele.

Původní návrh implementace EKE založené na DH metodě ustavení klíčů (tj. DHEKE) podporuje použití pevně dané hodnoty modulu β, která je buď z důvodu zvýšení rychlosti malá (několik set bitů) a nezamezuje těmto útokům, nebo je dostatečně velká (několik tisíc bitů) a za cenu snížení rychlosti jim zamezuje [BM92]. Protokol *DWEKE (dual-workfactor encrypted key exchange)* [Jas96] je vylepšenou variantou protokolu DHEKE a bez ztráty na rychlosti a efektivitě tomuto typu útoků zamezuje.

Základní myšlenka DWEKE spočívá v použití silnější varianty DHEKE s dostatečně velkými hodnotami modulu β pro ustavení hesla, zatímco pro standardní autentizační zprávy se i nadále používá z hlediska výkonu efektivnější DHEKE s malými hodnotami β. Návrh i popis celého protokolu je velmi silně ovlivněn snahou o co nejsnazší začlenění do systému Kerberos.

2.6 SPEKE

Protokol *SPEKE (simple password encrypted key exchange)* [Jab96] je svým návrhem a implementací velmi blízký protokolu DHEKE. I přes svou podobnost však mají tyto dva protokoly rozdílná omezení a nedostatky.

První fáze SPEKE je opět založena na DH metodě ustavení klíčů, ale namísto běžně používané fixní DH báze (generátoru) α, využívá SPEKE funkci f, která na základě svého jediného parametru (hesla) vytvoří nějakou bázi pro umocňování (tedy ne nutně generátor příslušné grupy). Výměna prvních dvou zpráv pak vypadá následovně:

$$\begin{array}{ccc} \boxed{\text{A}} & & \boxed{\text{B}} \\ f(P)^{r_A} \mod \beta & \longrightarrow & \\ & \longleftarrow & f(P)^{r_B} \mod \beta \\ & \ldots & \end{array}$$

Z předaných hodnot si pak nezávisle strana A i B vygeneruje klíč sezení například jako $K_S = f(P)^{r_A r_B} \mod \beta$ či $K_S = h(f(P)^{r_A r_B} \mod \beta)$. Funkci f je v případě použití bezpečného prvočísla $\beta = 2\gamma + 1$ doporučeno definovat jako $f(P) = P^{(\beta-1)/\gamma} \mod \beta = P^2 \mod \beta$ (tj. $f(P)$ stejně jako 2 je řádu γ), raději než $f(P) = 2^P \mod \beta$.

Protokol je tak odolnější, protože k případnému slovníkovému útoku již nestačí pouze jediný výpočet diskrétního logaritmu. Podrobnější informace vztahující se ke správné volbě f lze nalézt v [Jab96, Jab97].

V druhé fázi SPEKE pak obě strany opět ověřují zdali ustavení klíče sezení proběhlo korektně. Kromě náhodných čísel lze k tomuto účelu použít také kryptografické hašovací funkce. Strana A nejprve zašle straně B zprávu $h(h(K_S))$ a ta po jejím přijetí a ověření zašle straně A k ověření zprávu $h(K_S)$. Tento přístup je možný, protože klíč K_S vznikl z náhodných dat vygenerovaných oběma stranami a obsahuje také dostatek entropie.

SPEKE narozdíl od DHEKE v první fázi žádným způsobem nešifruje předávané zprávy, což útočníkovi dává možnost omezit prostor klíčů na malou množinu snadno předvídatelných hodnot – tzv. *subgroup confinement attack*. Podívejme se nyní na *man in the middle* verzi tohoto útoku popsanou v [vOW96]. Nechť δ je známý malý prvočíselný dělitel $\beta - 1$, pak útočník umocní předávané hodnoty na $(\beta - 1)/\delta$. Tím se z nich stanou generátory malé podgrupy řádu δ a útočník pak může klíč s pravděpodobností $1/\delta$ uhádnout nebo hrubou silou snadno nalézt. Útok lze také modifikovat tak, že se útočník vydává za jednu ze stran protokolu a druhé straně pošle přímo generátor podgrupy malého řádu.

Použití bezpečných prvočísel počet malých podgrup pouze redukuje a jako protiopatření by tedy mělo být vždy testováno, zdali výsledný klíč do těchto podgrup nepatří (nebo na základě jejich prvků nevznikl).

2.7 ASPEKE, BSPEKE a BEKE

Tato tři rozšíření protokolů jsou představena v [Jab97] a podobně jako AEKE zajišťují, že si server uchovává hesla pouze jednocestně zašifrována. ASPEKE je přímočará aplikace technik použitých k vytvoření AEKE na protokol SPEKE. BEKE a BSPEKE nahrazuje poslední část AEKE a ASPEKE dalším kolem DH metody ustavení klíčů, které umožňuje straně B ověřit, že strana A skutečně zná heslo.

Ověření znalosti hesla je u tohoto typu protokolů zcela nezbytné, protože jejich původní část zůstává až na použití jednocestně zašifrované verifikační hodnoty namísto otevřeného hesla naprosto beze změn (strana A si tuto verifikační hodnotu musí vždy z hesla dopočítat) a přímá znalost hesla tedy není prokázána – kdokoliv kdo zná verifikační hodnotu by mohl vystupovat za stranu A.

2.8 SRP

SRP (secure remote password) [Wu97] je zcela nový typ protokolu, který (stejně jako některé z předchozích protokolů) zajišťuje, že si server uchovává hesla pouze jednocestně zašifrována. Narozdíl od protokolů jako AEKE a ASPEKE (které jsou založeny na použití digitálních podpisů) či BEKE a BSPEKE (které využívají přidané kolo DH metody ustavení klíčů) je SRP založen na obecné konstrukci zvané *AKE (asymmetric key exchange)*. Tato konstrukce oproti EKE žádným způsobem nevyužívá symetrickou kryptografii, což činí výsledné protokoly jednodušší a mnohdy i bezpečnější (není již třeba řešit žádné problémy spjaté s používáním hesel jakžto

symetrických šifrovacích klíčů). Protokol SRP je speciální instancí AKE, a nabízí navíc vyšší výkon než srovnatelné protokoly jako například AEKE či BSPEKE.

Pro následující popis protokolu předpokládejme, že α je generátor multiplikativní grupy $\mathbb{Z}_\beta^*$, kde β je bezpečné prvočíslo, a že strana A na základě hesla P a soli[5] S vygeneruje veřejný klíč $x = h(P, S)$ a verifikační hodnotu $v = \alpha^x \mod \beta$. Hodnotu x pak smaže, zatímco hodnoty S, v doručí straně B (serveru).

V první části protokolu strana A zašle svůj identifikátor straně B, která jí nazpět zašle sůl S nezbytnou k výpočtu hodnoty x. Dále si strana A vygeneruje náhodné číslo $a \in \langle 2, \beta - 1\rangle$ a strana B náhodná čísla $b, u \in \langle 2, \beta - 1\rangle$. Vlastní popis druhé části protokolu pak vypadá následovně:

$$\begin{array}{cccc} \boxed{\text{A}} & & & \boxed{\text{B}} \\ A = \alpha^a \mod \beta & \longrightarrow & & \\ & \longleftarrow & & u,\ B = v + \alpha^b \mod \beta \\ & \ldots & & \end{array}$$

Strana A poté vypočítá společný klíč $K'_S = \alpha^{ab+bux} \mod \beta$ jako $(B - \alpha^x)^{a+ux} \mod \beta$ a strana B jako $(Av^u)^b \mod \beta$. Klíč sezení je pak vypočítán jako $K_S = h(K'_S)$ a jeho znalost si v závěrečné části protokolu opět obě strany ověří.

Bezpečnost této metody stojí na důkazu z [Wu97], že existence techniky vedoucí v polynomiálním čase k získání klíče sezení použitého v SRP, by vedla také v polynomiálním čase k rozbití DH metody ustavení klíčů. Z toho plyne, že SRP je alespoň tak bezpečný jako DH metoda ustavení klíčů.

Někdy je tento protokol nazýván SRP-3 a jeho vylepšená varianta SRP-6 [Wu02].

2.9 PDM

Posledním protokolem, který v této části práce popíšeme je *PDM (password derived moduli)* [PK01]. Tento protokol je opět založen na modifikaci DH metody ustavení klíčů, a jak již název napovídá, využívá heslo k vytvoření bezpečného prvočíselného modulu β (tj. modul musí být tvaru $\beta = 2\gamma + 1$, kde γ je velké prvočíslo).

Toho může být stranou A (klientem) dosaženo například využitím uživatelského hesla jako semínka generátoru pseudo-náhodných čísel, který bude k hledání odpovídající hodnoty β použit. Strana B (server) heslo nezná a má proto β jako verifikační hodnotu bezpečně uloženou. DH báze je u tohoto protokolu vždy $\alpha = 2$. Aby se při oboustranné autentizaci předešlo náročnému umocňování, tak má server navíc uloženy předpočítané hodnoty r_B a $2^{r_B} \mod \beta$. Celý protokol pak vypadá následovně:

$$\begin{array}{cccc} \boxed{\text{A}} & & & \boxed{\text{B}} \\ 2^{r_A} \mod \beta & \longrightarrow & & \\ & \longleftarrow & & 2^{r_B} \mod \beta, R, h(2^{r_B r_A} \mod \beta) \\ h(R, 2^{r_B r_A} \mod \beta) & \longrightarrow & & \end{array}$$

[5] Princip *solení* byl poprvé prezentován v [MT79] a aplikace této techniky v souvislosti s popisovanými protokoly je popsána například v [BM93, Jab97, Wu97]. Solení prakticky znemožňuje zjistit, zdali dvě (stejně) jednocestně zašifrovaná hesla vznikla ze stejného řetězce, a navíc útočníkovi brání vytvoření slovníku takto zašifrovaných hesel.

R je náhodně vygenerovaná hodnota. Při použití PDM výhradně k autentizaci mohou skutečně hodnoty r_B a $2^{r_B} \mod \beta$ zůstat pro daného klienta stejné. Pokud by ale cílem bylo i ustavení klíče sezení, bylo by nezbytné náhodné číslo r_B pro každý běh protokolu generovat znova (s čímž by souvisel i opětovný výpočet $2^{r_B} \mod \beta$).

Aby se předešlo redukci prostoru hesel, nesmí být žádná přenášená DH hodnota větší než jakákoliv β' (vytvořené na základě zkoušených hesel P'). Tento problém lze vyřešit zamítnutím použití takových náhodných čísel r_A a r_B, pro něž $2^{r_A} \mod \beta$ a $2^{r_B} \mod \beta$ jsou čísla větší, než nejmenší možná hodnota β vygenerovaná z nějakého hesla. Aby pravděpodobnost zamítání výše uvedených náhodných čísel byla co nejmenší (a předešlo se tak co nejvíce jejich opětovnému generování), bude se β volit z velmi úzkého intervalu prvočísel – v našem případě velmi blízkého mocnině dvou. Použijeme-li například 700bitová čísla, můžeme vhodný úzký interval získat fixním nastavením horních 64 bitů na binární hodnotu 1. Zbylý prostor 2^{636} čísel je pro vyhledávání bezpečných prvočísel stále dostatečně velký, a pravděpodobnost, že číslo $2^{r_A} \mod \beta$ či $2^{r_B} \mod \beta$ bude větší než nejmenší možná hodnota β je $1/2^{64}$.

V [PK99, PK01] je také uvedeno několik modifikovaných protokolů, které jsou určeny pro bezpečné stahování citlivých informací (například soukromých klíčů).

2.10 Shrnutí

V úvodu této kapitoly jsme se seznámili s protokolem EKE [BM92], který k ustavení klíče sezení využívá sdíleného hesla v kombinaci se symetrickou i asymetrickou kryptografií a poskytuje ochranu proti off-line útokům hrubou silou. Myšlenka tohoto zcela originálního protokolu se stala základem celé třídy nově vznikajících protokolů.

Oproti původnímu EKE zaručují protokoly DHEKE [BM92], DWEKE [Jas96], MEKE [STW95] či SPEKE [Jab96] navíc *dopřednou bezpečnost* (forward secrecy), což znamená, že kompromitace hesla neumožní útočníkovi získat klíče předcházejících sezení. Navíc začíná být také brána zřetel na to, aby případná kompromitace klíče sezení neumožňovala útoky vedoucí k získání hesla. Největší nevýhodou všech výše uvedených protokolů však stále zůstává nutnost uchovávat hesla na straně serveru v otevřené podobě. Tento nedostatek jako první překonává protokol AEKE [BM93], který je rozšířením EKE a umožňuje serveru ukládat hesla jednocestně zašifrována. Nevýhodou této modifikace EKE je, že protokol už nezaručuje dopřednou bezpečnost. Protokol BSPEKE [Jab97] již podobnými nedostatky za cenu podstatného zvýšení výpočetní složitosti netrpí. Efektivnější řešení pak nabízejí protokoly SRP [Wu97] a PDM [PK01].

Existuje samozřejmě mnohem větší množství protokolů založených na použití hesla, jejichž popis by ale již převyšoval rámec této práce. Z nejvýznamnějších zmiňme například protokoly jako AMP [Kwo00], AuthA [BR00], OKE [Luc97], PAK [Mac02], S3P [RCW98] či SNAPI [MSP00]. Velkou nevýhodou mnoha z těchto (a jim podobných) protokolů je, že nejsou prezentovány společně s důkazy, které by prokázaly jejich bezpečnost. Na některé z nich již byly objeveny útoky [Pat97].

Standardizací protokolů založených na použití hesla se zabývá pracovní skupina IEEE P1363 – viz draft IEEE P1363.2 [IEE05]. Cílem tohoto právě vznikajícího dokumentu však není upřednostnění některých technik či protokolů před jinými,

ale poskytnutí dostatečného množství různých metod, které se liší jak funkčností tak také efektivitou. Podrobný popis jednotlivých metod pak slouží jakožto návod k jejich implementaci (a to jak na straně serveru, tak také na straně klienta). Celkově se tento draft skládá z hlavní části (obsahující popis jednotlivých metod), příloh (poskytujících doprovodné informace) a nákresů jednotlivých postupů. Bohužel, jako mnoho jiných, je i tento draft prozatím značně nepřehledný a bez důkladné znalosti jednotlivých technik a protokolů téměř nečitelný.

Kapitola 3

Generování (pseudo)náhodných sekvencí

Zavedení eskalačních protokolů do praxe vyžaduje existenci kvalitních generátorů náhodných a pseudonáhodných sekvencí. V této kapitole shrneme na základě publikovaných materiálů informace vztahující se k metodám generování náhodných a pseudonáhodných sekvencí a hodnocení jejich kvality. Zaměříme se především na generátory vhodné pro kryptografické čipové karty, protože právě ty budou v následující kapitole předmětem našeho testování. Poté popíšeme princip vybraných generátorů pseudonáhodných sekvencí a u některých z nich také poukážeme na případné bezpečnostní nedostatky.

3.1 Úvod do problematiky

Abychom ujasnili námi používanou terminologii, definujme *generátor náhodných bitů* (RBG – random bit generator) jako zařízení či algoritmus jehož výstupem je *sekvence* statisticky nezávislých a neovlivněných binárních číslic [MvOV01]. Dále definujme (neformálně) *generátor pseudonáhodných bitů* (PRBG – pseudorandom bit generator), jako deterministický algoritmus, který ze vstupní náhodné binární sekvence délky k vygeneruje binární sekvenci $l \gg k$, která „bude vypadat“ náhodně – tj. bude v polynomiálním čase výpočetně nerozlišitelná od náhodné binární sekvence [MvOV01]. Vstup PRBG se nazývá *semínko* (seed).

Oba tyto generátory mohou být samozřejmě použity ke generování rovnoměrně rozložených *(pseudo)náhodných čísel*[1]. K získání takového čísla z intervalu $\langle 0, n\rangle$ je třeba vygenerovat (pseudo)náhodnou bitovou sekvenci o délce $\lfloor log_2 n \rfloor + 1$ a převést ji na odpovídající dekadický tvar. Pokud výsledek přesáhne n, můžeme jej například ignorovat a pokračovat vygenerováním nové (pseudo)náhodné bitové sekvence.

Generátor takovýchto (pseudo)náhodných čísel je pro čipovou kartu určenou k vykonávání kryptografických operací naprosto nezbytnou součástí a slouží například ke generování kryptografických klíčů, vytváření náhodných doplňujících hodnot či k implementaci algoritmických opatření, která zamezují útokům postranními kanály.

[1] Proto v dalším textu nebudeme rozlišovat mezi generátory náhodných bitů, sekvencí a čísel.

Výše uvedené základní dva typy generátorů se v komunitě vývojářů čipových karet také velmi často nazývají: *generátor náhodných čísel* (RNG – random number generator), někdy též označován jako *generátor skutečně náhodných čísel* (TRNG – true random number generator), a *generátor pseudonáhodných čísel* (PRNG – pseudorandom number generator). TRNG používá nedeterministický zdroj dat (resp. entropie) spolu s nějakou funkcí, která primární data zpracovává a vytváří náhodná data. Jeho výstup je možno použít buď přímo (v tomto případě je však nutno otestovat náhodnost generovaných čísel statistickými metodami), nebo jako vstup/semínko pro PRNG, který vytváří pseudonáhodná data. PRNG je v podstatě deterministický konečně stavový automat (FSM – finite state machine) a veškerá náhodnost jeho výstupu je tím redukována na vytvoření semínka (tj. entropie výstupu PRNG nebude nikdy vyšší než entropie semínka). Výhodou PRNG je, že jsou obvykle řádově rychlejší než TRNG, což pro některé aplikace může být podstatné.

3.1.1 Generátory náhodných čísel

Někteří výrobci začínají poslední dobou vybavovat standardní mikroprocesorové čipové karty dodatečnými obvody určenými k vykonávání specifických kryptografických operací. Jedním z těchto obvodů je i generátor skutečně náhodných čísel (tj. ne pouze pseudonáhodných).

Bohužel materiálů souvisejících s generátory skutečně náhodných čísel, které jsou navrženy k integraci do mikročipů, a jsou tedy vhodné pro čipové karty, je z důvodů utajení veřejně publikováno jen velmi málo. Obecně však lze podle [BBL04] rozlišit čtyři základní a často používané techniky generování skutečně náhodných čísel: přímé zesílení zdroje šumu (direct amplification of a noise source) [BB99, HCD97], vzorkování nestabilního oscilátoru (jittered oscillator sampling) [BGL+03, PC96, JK99], diskrétní chaotická mapování (discrete-time chaotic maps) [FCRV03, SK01] a metastabilní obvody [EHK+03, KC02, WF01].

Nepříjemnou vlastností hardwarových generátorů většinou je, že výsledná náhodná data mohu i přes precizní návrh generátoru stále vykazovat statistické závislosti (např. kvůli konstrukčním omezením, vnějším podmínkám, stáří apod.). V praxi se pak většinou využívá buď vzájemné kombinace různých metod generování (viz [PC00]) nebo speciálního digitálního obvodu, který případné statistické závislosti odstraní (ale také podstatně sníží rychlost generování náhodných dat).

Generátor náhodných čísel je například u čipových karet firmy Infineon (řady 66 a 88) založen na analogových dvojitých oscilátorech napětí. Tento typ generátorů náhodných čísel využívá frekvenční nestability volně běžícího oscilátoru, což je fenomén, který je pro generování náhodných čísel používán již od 50. let minulého století. Implementace popisovaná v [FMC84] využívá dvou oscilátorů (jeden generuje nižší frekvenci a druhý generuje frekvenci vyšší), jejichž výstup je digitálně mixován tak, že signál nižší frekvence je použit jako hodinový signál pro snímání úrovně signálu vyšší frekvence. Zlepšení charakteristik tohoto generátoru lze na úkor rychlosti dosáhnout použitím paritního filtru a obvodu promíchávajícího bity (scramble circuit).

Velkým implementačním problémem hardwarových generátorů náhodných čísel pro čipové karty je, že normálně běžné zdroje náhodných dat jsou jen velmi těžko

dostupné přímo v integrovaném obvodu. Častým řešením tohoto problému je použití analogových zdrojů entropie v digitálním obvodu a to i za cenu zvýšené spotřeby energie a potřebné plochy polovodičového čipu. Tyto analogové obvody mohou být navíc ovlivňovány periodickými signály v blízkosti generátoru. Vytvořením nenákladné plně digitální alternativy generátoru náhodných čísel pro použití v čipových kartách se zabývá [EHK+03]. Je popsán návrh integrovaného obvodu obsahujícího devět typů (a pro každý z nich ještě 15 až 31 variant s odlišným časováním) generátorů náhodných čísel, které využívají metastability bistabilního klopného obvodu. Ačkoliv se některé z těchto typů/variant takto realizovaných generátorů v určitých vnějších podmínkách ukázaly jako nepoužitelné (tj. produkovaly nenáhodný výstup), lze vzájemným XORováním jejich výstupů získat skutečně náhodná data (za předpokladu, že alespoň jeden z výstupů jsou skutečně náhodná data).

Vývojáři firmy Gemplus zase pro své čipové karty navrhli speciální čip Randaes, který mimo jiné obsahuje tři experimentální hardwarové generátory náhodných čísel založené na zesilování zdroje šumu (analogový RNGa) a vzorkování nestabilního oscilátoru (analogový RNGb a plně digitální RNGc). Všechny tyto generátory používají stejný adaptivní digitální dekorelátor, který ve výsledných sekvencích potlačuje případné statistické závislosti a zaručuje tak jejich korektnost bez další statistické analýzy výstupu. Podrobný popis je obsažen v [BTSL01]. Dále také navrhli plně digitální vysokorychlostní generátor náhodných čísel, založený opět na použití vzorkování nestabilního oscilátoru a vhodný pro implementaci do čipových karet. Podrobný popis je obsažen v [BGL+03].

Zajímavý návrh plně digitálního TRNG, který je opět založen na vzorkování nestabilního oscilátoru, je popsán v [BBL04]. Přidáním zpětnovazebného cyklu lze navíc pomocí precizně navrženého offsetu dosáhnout metastability bistabilního klopného obvodu, a získat tak další velmi kvalitní zdroj náhodných dat.

V [Tka02] je popsán zcela odlišný typ generátoru skutečně náhodných čísel, který kombinuje prvky TRNG i PRNG (používá dvou volně běžících oscilátorů k časování dvou konečně stavových automatů). Tento generátor však není příliš kvalitní (resp. bezpečný) – v [Dic03] je na něj popsán teoretický útok.

3.1.2 Generátory pseudonáhodných čísel

Protože generátor pseudonáhodných čísel je v podstatě deterministický konečný automat, nečiní jeho implementace (ať již hardwarová či softwarová) do kryptografických čipových karet příliš velké problémy. Nejpodstatnějšími vlastnostmi PRNG jsou kromě bezpečnosti (ve smyslu nepředvídatelnosti) také rychlost a u softwarové implementace paměťová náročnost. Základní generátory pseudonáhodných čísel jsou popsány například v [MvOV01].

K velmi často používaným (především pro účely simulací a tvorbu náhodnostních algoritmů) patří lineární kongruenční generátor (LCG – linear congruential generator) [MvOV01]. Jeho výstup je však předvídatelný, a proto je tento typ PRNG pro kryptografické účely zcela nevhodný [Boy89, Kra90].

Další a asi nejjednodušší konstrukce generátoru pseudonáhodných čísel je založena na posuvných registrech s lineární zpětnou vazbou (LFSR – linear feedback shift register). Tento typ PRNG je velmi efektivně implementovatelný v hardware,

dokáže vytvářet dlouhé pseudonáhodné sekvence s dobrým statistickým rozložením a jeho vlastnosti jsou dobře teoreticky zmapovány. Protože však sám o sobě není také příliš bezpečný, slouží typicky jako základní blok k vytváření složitějších pseudonáhodných generátorů. Toho lze dosáhnout buď nelineární kombinací několika LFSR, nebo použitím jednoho (či kombinace více) LFSR k časování druhého (či kombinace více) LFSR. Více informací o LFSR lze nalézt v [Koe, Neu04, Zen04].

Zcela odlišnou třídu generátorů pseudonáhodných čísel tvoří kryptograficky bezpečné generátory, založené na obtížných problémech teorie čísel. Modulární aritmetika, kterou tyto generátory využívají, je sice činí velmi pomalými, ale i přesto mohou být efektivně využity v zařízeních s dodatečnou hardwarovou podporu. V [MvOV01] jsou popsány tři generátory spadající do této kategorie: RSA PRNG, Micali-Schnorr PRNG a Blum Blum Shub (BBS) PRNG. První dva jsou založeny na známém problému faktorizace čísel na němž stojí i RSA, zatím co BBS PRNG využívá problému kvadratických zbytků. V [PKS00] je popsána také rychlejší verze BBS PRNG, která k zvýšení efektivity používá Montgomeryho násobení (to je v současné době hardwarově akcelerováno v kryptografických čipových kartách mnoha výrobců).

Mezi nejrychlejší PRNG v této třídě patří generátor založený na problému diskrétního logaritmu s krátkými (resp. malými) exponenty. Tento PRNG v podstatě pouze opakovaně umocňuje pevně danou bázi na krátký exponent, přičemž během každé iterace produkuje několik bitů jako výstup a zbytek používá jako exponent pro další iteraci. Jeho podrobný popis lze nalézt v [Gen00]. Na podobném principu funguje i efektivní a implementačně jednoduchý PRNG, který je popsán v [DRV02]. Oproti předchozímu je tento generátor založen na problému faktorizace čísel, a je pomalejší.

V praxi velmi často používané generátory pseudonáhodných čísel jsou ale většinou založeny na použití běžných kryptografických funkcí. Jediné standardizované generátory jsou ANSI X9.17 PRNG [ANS85] a FIPS 186 PRNG [FIP94b]. ANSI X9.17 PRNG je založen na použití dvouklíčového 3DES a generátory popsané v normě FIPS 186 využívají DES nebo hašovací funkci SHA-1. Mimo oficiální dokumenty lze jejich popis nalézt v [MvOV01, KSWH98]. Podrobnou bezpečnostní analýzou a efektivnějšími možnostmi použití především těchto standardizovaných generátorů se zabývá [DHY02]. Pozornost je zde věnována zejména vstupům generátorů, jejich stavům a použitým kryptografickým funkcím. Často používané jsou ale i nestandardizované generátory implementované v RSAREF 2.0 [Lab94] a Cryptolib [LMS93].

Běžných kryptografických funkcí využívá také PRNG Yarrow-160. Originální návrh je sice postaven na tříklíčové variantě 3DES a na SHA-1, ale v principu nic nebrání použití AES spolu s SHA-2. Velmi důmyslný návrh tohoto generátoru jej činí dobře odolným proti všem známým útokům (zejména těm popsaným v [KSWH98]). Jeho podrobný popis je obsažen v [KSF99].

V [ARV95] je představen generátor pseudonáhodných čísel používající DES. Jeho efektivní a především paměťově nenáročná softwarová implementace je popsána v [ARV99]. Tato implementace si oproti původním 16-32KB pamětí vystačí s pouhými 3KB paměti a je proto vhodná zejména pro použití v kryptografických čipových kartách.

Nejnovějším z popisovaných PRNG je pak generátor Cilia [Ng05], který kromě hašovacích funkcí používá také dvojité šifrování.

Útoky na generátory pseudonáhodných čísel

V [KSWH98] jsou PRNG posuzovány z pohledu útočníka: jsou prezentovány požadavky na PRNG, základní funkční model ideálního PRNG, a výčet útoků na ideální i skutečně používané PRNG (např. ANSI X9.17). Speciálně jsou zvažovány způsoby, kterými může útočník zapříčinit že PRNG již neprodukuje náhodná data, či které umožňují využít některé z výstupů (jako například inicializačních vektorů) k uhádnutí jiných výstupů (jako například klíčů sezení). Útoky jsou zde rozděleny do tří kategorií: přímé kryptoanalytické útoky, útoky na vstup a útoky založené na kompromitaci tajného stavu PRNG. V závěrečné části jsou pak také navržena možná protiopatření a rady týkající se správného návrhu bezpečných PRNG.

Analýzou a útoky na PRNG založené na použití LFSR se zabývá disertační práce [Zen04]. Cílem práce je ale i přesto návrh bezpečných PRNG, a kryptoanalýza zde má spíše konstruktivní charakter – jedině lepší porozumění všem možným problémům a útokům může vést k novým kritériím správného návrhu bezpečných PRNG. Jsou popsány techniky kryptoanalýzy jak proti PRNG jejichž specifikace je známa, tak také proti neznámým PRNG.

3.1.3 Testování náhodnosti

Náhodnost je pravděpodobnostní vlastnost, a existuje tedy i poměrně velké množství statistických testů, hodnotících zdali je či není daná sekvence náhodná. Žádná konečná množina takovýchto testů však nemůže být považována za zcela úplnou, a proto je třeba interpretovat výsledky statistických testů s jistým nadhledem. Testování se většinou týká jak TRNG, tak také PRNG.

Jak uvádí [Sot00], existuje pět základních zdrojů pojednávajících o testech náhodnosti. Jsou to knihy [MvOV01, Knu97] a sady testů DIEHARD [Mar], Crypt-XS [Cry] a NIST [FIP00]. V těchto zdrojích jsou popsány desítky různých statistických testů náhodnosti. Podle [HSS04] jsou ale již některé testy (např. z [Knu97]) zastaralé, protože jimi snadno projde i mnoho evidentně slabých generátorů. Sada testů DIEHARD byla naopak dlouhou dobu považována za nejlepší/nejúplnější, ale v [MT02] již vyšla nová sada tří testů, o níž se tvrdí, že je ještě lepší než DIEHARD – tj. ty generátory, které projdou touto sadou testů, projdou i testy DIEHARD (a navíc je jejich implementace mnohem snazší). Podle [SJUH04] jsou drobné korekce v nastavení testů nutné dokonce i pro sadu testů NIST. V [HSS04] je také popsán nový moderní náhodnostní test SAC (Strict Avalanche Criterion), který je schopen velmi dobře detekovat špatné vlastnosti výstupní sekvence a nevyžaduje příliš mnoho výpočetního výkonu ani času.

Standard FIPS 140-1/2 [FIP94a, FIP01] doporučuje spouštění testů náhodnosti generovaného výstupu po každém resetu karty. Pouze po úspěšném proběhnutí těchto testů je možné poskytovat (pseudo)náhodná čísla aplikaci na kartě. Ve skutečnosti však jen velmi málo kryptografických čipových karet takovéto testy provádí.

Zcela odlišnou kategorii testů tvoří online testy pro TRNG [Sch01], které přímo za běhu TRNG zaručují, že vygenerovaná data jsou skutečně náhodná. Základními požadavky na online testy (zvláště pokud je příslušný TRNG implementován na čipové kartě) je rychlost, paměťová nenáročnost a snadnost implementace (tj. pouze několik řádků kódu).

Statistickým testováním náhodnosti se budeme podrobněji zabývat v kapitole 4.

3.2 Popis vybraných generátorů

Protože detailní popis funkčnosti generátorů skutečně náhodných sekvencí značně převyšuje rámec této práce (a nespadá ani do oblasti informatiky), budeme se v této části zabývat výhradně generátory pseudonáhodných sekvencí. Připomeňme, že tyto generátory musí být schopny na požádání generovat pseudonáhodné sekvence, které budou v polynomiálním čase výpočetně nerozlišitelné od skutečně náhodných sekvencí. Vycházet budeme především z [KSWH98, MvOV01].

Funkce $LSB_k()$ bude vracet k nejméně významných bitů zadané vstupní hodnoty, funkce $MSB_k()$ bude vracet k nejvíce významných bitů zadané vstupní hodnoty a symbol $\leftarrow$ bude značit přiřazení.

3.2.1 RSA PRNG

Tak jako kryptosystém RSA stojí i tento kryptograficky bezpečný PRNG na známém problému faktorizace čísel. Podobně jako u RSA, je nejprve v inicializační fázi nutno vygenerovat dvě velká prvočísla p, q, na jejich základě vypočítat hodnoty $n = pq$, $\phi(n) = (p-1)(q-1)$ a zvolit náhodné číslo e takové, že $1 < e < \phi(n)$ a $gcd(e, \phi(n)) = 1$. Poté je z intervalu $\langle 1, n-1\rangle$ vybráno náhodné číslo x_0, které tvoří vstup (semínko) PRNG. Samotný proces generování pak probíhá tak, že se v cyklu pro i od 1 do l provádějí následující dvě operace: $x_i \leftarrow x_{i-1}^e \mod n$; $z_i \leftarrow LSB_1(x_i)$. Výstupem tohoto generátoru je sekvence l bitů $z_1, z_2, \ldots, z_l$.

3.2.2 Micali-Schnorr PRNG

Tento kryptograficky bezpečný PRNG je efektivnější modifikací RSA PRNG. V inicializační fázi je opět nutno vygenerovat dvě velká prvočísla p, q a na jejich základě vypočítat hodnoty $n = pq$, $\phi(n) = (p-1)(q-1)$. Dále $N = \lfloor log_2 n \rfloor + 1$ je bitová délka n a náhodné číslo e se volí takové, že $1 < e < \phi(n)$, $gcd(e, \phi(n)) = 1$ a $80e < N$. Nechť $k = \lfloor 1 - 2/e \rfloor$ a $r = N - k$. Semínko x_0 se volí, aby mělo bitovou délku právě r. Samotný proces generování pak probíhá tak, že se v cyklu pro i od 1 do l provádějí následující tři operace: $y_i \leftarrow x_{i-1}^e \mod n$; $x_i \leftarrow MSB_r(y_i)$; $z_i \leftarrow LSB_k(y_i)$. Výstupem tohoto generátoru je sekvence kl bitů $z_1 \| z_2 \| \ldots \| z_l$, kde $\|$ značí zřetězení.

3.2.3 Blum Blum Shub PRNG

Kryptograficky bezpečný BBS PRNG využívá problému kvadratických zbytků, ale jak bylo později dokázáno, je bezpečný i za slabšího předpokladu, že faktorizace dvou čísel je obecně v polynomiálním čase neřešitelná. V inicializační fázi je nutno vygenerovat dvě rozdílná velká prvočísla p, q kongruentní $3 \mod 4$ (tzv. *Blumova prvočísla*) a na jejich základě vypočítat hodnotu $n = pq$. Poté je z intervalu $\langle 1, n-1\rangle$ vybráno náhodné číslo s takové, že $gcd(s, n) = 1$. Na jeho základě je spočítána hodnota $x_0 \leftarrow s^2 \mod n$ a samotný proces generování pak probíhá tak, že se v cyklu pro i od 1 do l provádějí následující dvě operace: $x_i \leftarrow x_{i-1}^2 \mod n$; $z_i \leftarrow LSB_1(x_i)$. Výstupem tohoto generátoru je sekvence l bitů $z_1, z_2, \ldots, z_l$.

3.2.4 ANSI X9.17 PRNG

Velmi často používaný ANSI X9.17 PRNG je založen na použití dvouklíčového 3DES-2 v režimu encrypt-decrypt-encrypt (EDE). Původně byl navržen k vytváření klíčů algoritmu DES a inicializačních vektorů (IV). Vstupem je náhodné tajné 64bitové semínko s, číslo l a šifrovací klíč K. Nejprve je vytvořena hodnota $I = E_K(D)$, kde D je 64bitová reprezentace data a času. Samotný proces generování pak probíhá tak, že se v cyklu pro i od 1 do l provádějí následující dvě operace: $z_i \leftarrow E_K(I \oplus s)$; $s \leftarrow E_K(z_i \oplus I)$. Výstupem tohoto generátoru je sekvence l 64bitových řetězců $z_1, z_2, \ldots, z_l$. Má-li některý z nich být použit jako klíč algoritmu DES, musí být po vygenerování ještě upraven, aby byla zajištěna požadovaná lichá parita.

Oproti této chybné verzi ANSI X9.17 PRNG z [MvOV01] by ale podle [FIP02, KSWH98, Kel05] mělo být I vytvořeno při každé iteraci vždy znova, což také ve zbytku práce budeme předpokládat.

Poznamenejme dále, že existuje skutečně mnoho modifikovaných verzí a implementací tohoto generátoru. Některé z nich (viz [Dai97]) vytvářejí hodnotu I_i jako $I_i \leftarrow E_K(I_{i-1} \oplus D)$, což odpovídá šifrování pomocí modu CBC namísto původního ECB, jiné (viz [FIP02]) namísto reprezentace data a času D umožňují použít čítač, který musí každou iteraci měnit svou hodnotu. Namísto 3DES lze podle NIST nejnověji použít i AES [Kel05]. Další rozdíly v implementacích mohou být také způsobeny možností v podstatě kdykoliv (re)generovat tajný klíč K.

3.2.5 FIPS 186 PRNG

FIPS 186 PRNG byl původně navržen pro generování pseudonáhodných sekvencí pro DSA (Digital Signature Algorithm). Využívá hašovací funkci SHA-1 nebo alternativně konstrukci založenou na DES (v dalším označovány jako h). Vstupem algoritmu je celočíselná hodnota l a 160bitové prvočíslo q. Při použití pro generování klíčů povoluje volitelný uživatelský vstup w_i, který však není povolen při generování ostatních parametrů DSA (kdy je nastaven na nulu). Pokud není uvedeno jinak, je veškerá aritmetika v tomto generátoru prováděna modulo 2^n, kde $160 \leq n \leq 512$ je pevně zvoleno. Tajné semínko s je vygenerováno tak, aby mělo bitovou délku právě n. Samotný proces generování pak probíhá tak, že se v cyklu pro i od 1 do l provádějí následující dvě operace: $z_i \leftarrow h((w_i + s) \mod 2^n) \mod q$; $s \leftarrow s + z_i + 1 \mod 2^n$. Výstupem tohoto generátoru je sekvence l pseudonáhodných čísel $z_1, z_2, \ldots, z_l$ z intervalu $\langle 0, q-1 \rangle$.

3.2.6 RSAREF 2.0 PRNG

Tento PRNG je založen na používání pouze dvou operací – hašování pomocí MD5 (funkce h) a sčítání modulo 2^{128}. Oproti předchozím generátorům, které po zpracování každého vstupu vždy generují nějaký výstup, jsou u RSAREF 2.0 PRNG vstupní data zpracovávána podle dostupnosti. Generátor si uchovává 128bitový čítač c_i. Vstupy s_i zpracovává příkazem: $c_{i+1} \leftarrow c_i + h(s_i) \mod 2^{128}$. Metoda pro generování výstupů z_i je pak složena ze dvou operací: $z_i \leftarrow h(c_i) \mod 2^{128}$; $c_{i+1} \leftarrow c_i + 1 \mod 2^{128}$.

3.3 Kryptoanalytické útoky na PRNG

V této části práce se zaměříme na kryptoanalýzu několika generátorů pseudonáhodných sekvencí, které jsou založeny na použití běžných kryptografických funkcí. Zabývat se budeme především ANSI X9.17 PRNG, FIPS 186 PRNG a RSAREF 2.0 PRNG. Vycházet zde budeme z [KSWH98].

Všechny tyto generátory si udržují tajný stav S, který by neměl být útočníkem nijak zjistitelný. Tento stav se mění na základě útočníkovi neznámých vstupních hodnot, které jsou získávány z nejrůznějších fyzikálních procesů. Generátory většinou startují ze známého (či snadno zjistitelného) stavu a až po zpracování velkého množství vstupů dosáhnou bezpečného stavu. V praxi je velice prozíravé předpokládat, že občasná kompromitace stavu může nastat – i v tomto případě by však dobře navržený PRNG měl zůstat bezpečný a útočník by neměl mít možnost získaného stavu zneužít. Poznamenejme dále, že v principu jakákoliv metoda umožňující útočníkovi rozlišit mezi výstupem PRNG a skutečně náhodným výstupem je považována za útok. V praxi ale útočníka samozřejmě mnohem více zajímá správná předpověď výstupů, které teprve budou generovány. Celkem rozlišujeme tři kategorie útoků:

Přímé kryptoanalytické útoky – jsou takové útoky, jejichž pomocí je útočník schopen přímo rozlišit výstup PRNG od skutečně náhodného výstupu. Tento druh útoků je aplikovatelný na většinu (ale ne na všechny) PRNG. Je-li totiž PRNG určen ke generování výstupů, které nemohou být útočníkem přímo sledovány (např. tajné klíče), nebude tímto útokem nikdy zranitelný.

Útoky na vstup – umožňují útočníkovi na základě znalosti či ovlivňování vstupů PRNG rozlišit jeho výstup od skutečně náhodného výstupu. Můžeme je rozdělit na útoky *známého vstupu* (known-input), *zvoleného vstupu* (chosen-input) a *přehraného vstupu* (replayed-input). Útoky využívající známého vstupu jsou samozřejmě reálné především v situacích kdy lze vstup snadno předpovědět. Typickým příkladem může být aplikace, která jakožto vstup pro PRNG využívá doby zpoždění (prodlevy) pevného disku, ale je spuštěna tak, že používá síťové disky jejichž doby zpoždění jsou útočníkem snadno pozorovatelné. Útoky využívající zvoleného vstupu jsou velmi účinné proti čipovým kartám, tokenům či podobným zařízením odolným vůči narušení. Mohou být ale také snadno použitelné proti aplikacím, které získávají vstupy pro PRNG ze síťových statistik, přicházejících zpráv či jiných snadno ovlivnitelných zdrojů entropie. Útoky využívající přehraného vstupu lze aplikovat v podobných situacích jako útoky využívající zvoleného vstupu.

Útoky založené na kompromitaci tajného stavu – se snaží co nejdelší dobu využít dříve získaného stavu PRNG. Předpokládáme, že se útočníkovi nějakým způsobem podařilo v určitém okamžiku zjistit tajný stav PRNG. Útok je úspěšný, pokud umožňuje obnovit neznámé výstupy PRNG (nebo je rozlišit od skutečně náhodných výstupů) z doby před nebo po kompromitaci stavu. Tyto útoky často využívají nedostatku entropie při startu generátoru nebo kompromitace stavu jedním z dále uvedených útoků, které můžeme rozdělit na *zpětné útoky* (backtracking attacks), *trvale kompromitující útoky* (permanent

compromise attacks), *útoky opakovaným hádáním* (iterative guessing attacks) a *útoky setkání uprostřed* (meet in the middle attacks). Zpětné útoky využívají stavu S kompromitovaného v určitém čase t k získání předcházejících stavů (a výstupů) PRNG. Trvale kompromitující útoky využívají stavu S kompromitovaného v určitém čase t k získání všech předchozích i budoucích stavů (a výstupů) PRNG. Útoky opakovaným hádáním využívají stavu S kompromitovaného v určitém čase t k získání hodnoty stavu (a výstupu) v čase $t + \epsilon$ (pouze v případě kdy lze odhadnout vstupy generátoru mezi časovými okamžiky t a $t+\epsilon$). Útoky setkání uprostřed využívají stavu S kompromitovaného v čase t a $t + 2\epsilon$ k získání hodnoty stavu (a výstupu) v čase $t + \epsilon$.

3.3.1 ANSI X9.17 PRNG

Přímý kryptoanalytický útok na tento generátor by vyžadoval úspěšnou kryptoanalýzu 3DES (která nebyla nikdy veřejně prokázána). Znalost či ovlivňování vstupů také nečiní generátor pro útočníka zranitelným. Z kompromitace stavu se však již ANSI X9.17 PRNG řádně nezotaví. Útočník, který získá klíč K (součást vnitřního stavu generátoru) může bez větší námahy kdykoliv kompromitovat celý vnitřní stav.

Trvale kompromitující útok spočívá v odvození interního stavu generátoru z jeho dvou po sobě jdoucích výstupů. Předpokládáme pouze znalost klíče K a výstupů z_i a z_{i+1}. Útočníkovým cílem je získání hodnoty s_{i+1} (je-li vnitřní tělo cyklu zapsáno s indexy a v souladu z [KSWH98] jako: $I_i \leftarrow E_K(D)$; $z_i \leftarrow E_K(I_i \oplus s_i)$; $s_{i+1} \leftarrow E_K(z_i \oplus I_i)$). Toho lze triviálně dosáhnout úplným prohledáváním 64bitového prostoru. Mnohdy lze ale předpokládat, že každá 64bitová reprezentace data a času obsahuje pouze 10 bitů, které nejsou ještě útočníkovi známy (u hodin s přesností milisekund, je útočník schopen přibližně určit sekundu kdy byl výstup vygenerován). Tím lze útok redukovat tak, že vyžaduje asi 2^{11} šifrování klíčem K. Platí totiž, že $s_{i+1} = D_K(z_{i+1}) \oplus I_{i+1}$ a zároveň $s_{i+1} = E_K(z_i \oplus I_i)$. Útočník pak jen vyzkouší všechny přípustné hodnoty pro I_i a I_{i+1}, čímž získá dva seznamy kandidátů na s_{i+1}. Správná hodnota s_{i+1} je jediná, která se nachází zároveň v obou seznamech.

Útok opakovaným hádáním umožňuje útočníkovi, který zná s_i, K a nějakou funkci výstupu z_{i+1}, odhalit hodnotu s_{i+1}. Jeho příčinou je opět nedostatek entropie v 64bitové reprezentaci data a času. Oproti výše uvedenému trvale kompromitujícímu útoku však stačí znát pouze nějakou funkci výstupu (a ne přímo výstup samotný). To může být například zpráva zašifrovaná klíčem odvozeným z výstupu. Zpětný útok funguje v podstatě na stejném principu.

Útok setkání uprostřed je kombinací předcházejících dvou útoků. Lze použít v případech, kdy je generováno velké množství náhodných dat, která nejsou přímo pozorovatelná. Můžeme například znát hodnoty s_i a s_{i+8}, ale žádné z dalších mezilehlých hodnot. Útočník pouze na základě hádání a konstrukce sekvencí $I_{i+1} \ldots I_{i+4}$ a $I_{i+5} \ldots I_{i+8}$ vytvoří v čase 2^{41} dva seznamy možných hodnot s_{i+4}. Hledaná hodnota s_{i+4} bude přítomna v obou seznamech a její pomocí získá správnou sekvenci $I_{i+1} \ldots I_{i+8}$ (a samozřejmě i výstupy generátoru). Protože se však v seznamech může vyskytovat až 2^{16} stejných hodnot (kandidátů na s_{i+4}), je potřeba ještě určit tu správnou z nich. Bylo-li například všech 8 výstupních bloků použito jako šifrovací klíč k zašifrování nějaké zprávy, lze toho dosáhnout pokusy o její dešifrování.

3.3.2 FIPS 186 PRNG

Budeme se zabývat nejslabší možnou variantou FIPS 186 PRNG, kdy jsou povoleny uživatelské vstupy w_i a hodnota n je 160. Přímý kryptoanalytický útok na tento generátor by závisel na kvalitě použité hašovací funkce. Je-li hašovací funkce dostatečně kvalitní, budou výstupy PRNG jen velmi těžko rozlišitelné od skutečně náhodných sekvencí. Nicméně žádná analýza kvality výstupu PRNG v závislosti na použitých hašovacích funkcích není doposud známa. Oproti tomu, ovlivňování vstupní hodnoty w_i (pokud je zasílána přímo) umožňuje přinutit generátor produkovat stále stejný výstup. Stačí jen nastavit $w_i = w_{i-1} - z_{i-1} - 1 \mod 2^{160}$ (vnitřní tělo cyklu je s indexy zapsáno jako: $z_i \leftarrow h((w_i + s_i) \mod 2^{160}) \mod q$; $s_{i+1} \leftarrow s_i + z_i + 1 \mod 2^{160}$). Pokud jde o útoky založené na kompromitaci stavu, je v tomto ohledu FIPS 186 PRNG lepší než ANSI X9.17 PRNG.

Pro trvale kompromitující útok předpokládejme, že útočník v nějakém okamžiku kompromitoval stav generátoru. FIPS 186 PRNG sice používá vstupy při vytváření výstupů, ale narozdíl od ANSI X9.17 PRNG, pokud útočník vidí výstupy generátoru, tak se nemusí vstupy pokoušet uhádnout. Chybou FIPS 186 PRNG totiž je, že hodnota s_{i+1} nezávisí na w_i přímo. Pokud by tomu tak bylo, mohl by útočník na základě znalosti stavu stále zkoušet hodnotu vstupu uhádnout, ale v případě neúspěchu by znalost stavu ztratil.

Útok opakovaným hádáním také zneužívá kompromitovaného stavu. Pokud totiž útočník zná hodnotu s_i a navíc ví, že w_i obsahuje pouze 20 bitů entropie, může vytvořit seznam obsahující 2^{20} 160bitových výstupů (z nichž právě jeden je ten správný). Má-li dále k dispozici nějakou funkci hledaného výstupu PRNG (například zprávu podepsanou algoritmem DSA), je již jeho nalezení snadné. Poznamenejme ještě, že znalost s_i a nalezení výstupu z_i umožňuje ihned určit hodnotu s_{i+1}.

Zpětný útok pak umožňuje z s_i a z_{i-1} získat ihned hodnotu s_{i-1}, protože $s_{i-1} = s_i - z_{i-1} - 1 \mod 2^{160}$. Podobně varianta útoku setkání uprostřed umožňuje se znalostí s_i, s_{i+2} a z_{i+1} získat hodnotu z_i. Stačí si jen uvědomit, že $z_i = s_{i+2} - s_i - z_{i+1} - 2 \mod 2^{160}$.

3.3.3 RSAREF 2.0 PRNG

I přesto, že kvůli možnosti efektivního hledání kolizí již není MD5 považována za bezpečnou [Kat05, Len05], není jasné, jak toho využít k přímému kryptoanalytickému útoku na tento generátor. Podobně je v podstatě nepoužitelné předpočítání některých čítačů a jejich příslušných hašů. Zvyšování hodnoty 128bitového čítače však umožňuje časovou analýzu, která spolu s útoky založenými na ovlivňování vstupních hodnot činí generátor poměrně zranitelný. Tyto útoky ale již převyšují rámec práce a jejich podrobný popis lze nalézt v [KSWH98].

Generátor je také zranitelný útoky založenými na kompromitaci tajného stavu. Mechanizmus zpracovávání vstupů je potenciálně nebezpečný, protože nezohledňuje pořadí vstupů. Tím pádem je pravděpodobnější, že generátor začne pracovat v nezabezpečeném stavu, a že bude trvat mnohem déle, než se do bezpečného stavu dostane. Útok opakovaným hádáním umožňuje se znalostí c_i a snadno uhádnutelnou hodnotou s_i vygenerovat výstup z_{i+1} (útočník jej bude vždy znát) a udržet si i nadále znalost stavu PRNG. Podobně funguje i zpětný útok.

3.3.4 Shrnutí

Generátory pseudonáhodných čísel tvoří specifickou třídu kryptografických primitiv, a tak bychom se k nim také měli stavět. Vytváření abstraktních útoků na idealizované PRNG a jejich ověřování na reálných PRNG by mělo být naprosto běžné (stejně jako v případě blokových šifer, proudových šifer či kryptografických hašovacích funkcí). Nové PRNG by pak měly být navrhovány s ohledem na existující útoky takovým způsobem, aby jim zcela zamezily či (tam kde to není možné) alespoň co nejvíce odolávaly.

Mnohé současně používané PRNG mohou být s ohledem na existující útoky za určitých podmínek používány i nadále, ale některým útokům lze zabránit jen výraznou změnou celého procesu generování. Zajímavé návrhy protiopatření jsou popsány v [KSWH98], připomeňme však, že vhodnými a efektivními metodami použití ANSI X9.17 PRNG a FIPS 186 PRNG se zabývá také [DHY02].

Kapitola 4

Testování TRNG

V této kapitole se budeme zabývat testy generátorů skutečně náhodných sekvencí implementovaných na vybraných kryptografických čipových kartách. Nejprve se seznámíme s použitým vybavením, které je nezbytné k získání skutečně náhodných sekvencí z čipových karet. Poté se budeme zabývat problematikou statistického testování náhodnosti těchto sekvencí.

4.1 Použité vybavení

Pro praktickou část diplomové práce jsme měli k dispozici vývojářskou soupravu GemXpresso RAD III Kit od firmy Gemplus. Ta je určena k návrhu a testování nových aplikací pro kryptografické čipové karty GemXpresso s podporou technologie JavaCard™. Součástí této soupravy je čtečka čipových karet, sada několika druhů kryptografických čipových karet a mnohé vývojové softwarové nástroje, jako například:

- JCardManager – umožňuje zasílání různých APDU (application protocol data unit) příkazů, nahrávání apletů a kompletní správu čipové karty.
- GemXpresso SmartCards Simulator – simuluje prostředí vybrané čipové karty, může být použit k testování a debugování apletů.
- Plugin do JBuilderu a Visual Cafe – zajišťuje například snadné vytváření JavaCard projektů či snadné spouštění konverzí apletů pro různé typy čipových karet.

Software GemXpresso RAD III je plně kompatibilní se standardy Java Card 2.1, Open Platform 2.0.1, Open Card Framework nebo PCSC. Z operačních systémů je kompatibilní s Windows 98, Windows NT Workstation a Windows 2000, ale problémy nečinila ani práce ve Windows XP Professional. K programování potřebných apletů jsme využili JBuilder 6.0 Enterprise. Testování jsme podrobili čipové karty GXP211 PK, GXP211 PK IS, GXPLite-Generic, GXPPro R3 (některé z nich však nebyly součástí vývojářské soupravy).

To co nás z pohledu použitého vybavení zajímalo samozřejmě nejvíce, byly typy generátorů skutečně náhodných čísel implementovaných na jednotlivých čipových

kartách. Bohužel právě tato informace je výrobci čipových karet před veřejností utajována. Po osobní e-mailové korespondenci [Smo05] jsme zjistili, že firma Gemplus často používá čipy vyráběné firmou Infineon, jejichž TRNG bývají typicky založeny na analogových dvojitých oscilátorech napětí. Je-li však výstup tohoto typu TRNG na různých čipových kartách nějakým způsobem dále upravován se nám již zjistit nepodařilo. Pro některé druhy karet využívá Gemplus také čipy vyráběné firmami Atmel, Philips a STMicroelectronics [VIS05]. O jejich generátorech skutečně náhodných čísel jsme se také žádné informace nedozvěděli.

4.1.1 Získávání skutečně náhodných sekvencí

Před samotným testováním jsme nejprve museli z čipových karet vygenerovat dostatečné množství skutečně náhodných dat (původně 128MB dat, později jsme se ale spokojili jen s 50MB). S využitím JavaCard™ API [Sun99] byl naprogramován aplet, který po nahrání do příslušných karet zajišťoval generování skutečně náhodných sekvencí. Bohužel JCardManager umožňuje zasílání jednotlivých APDU pouze ručně, a proto jsme k zasílání a zpracování velkého množství APDU využili aplikaci SCTool naprogramovanou Petrem Švendou.

Poznamenejme, že proces generování tak obrovského množství náhodných dat byl s výjimkou karty GXPLite-Generic velmi pomalý – u některých karet zabral až 14 dní, u jiných se zase na první pokus nepodařilo požadované množství dat vůbec vygenerovat. Kartě GXPLite-Generic nedělalo problémy za velmi krátkou dobu vygenerovat dokonce 1GB náhodných dat.

4.2 Statistické testování náhodnosti

Celá problematika testování náhodnosti je založena na testování statistických hypotéz. Vycházíme zde především z [FIP00, BMO01, Nov99]. *Nulová hypotéza* H_0 značí testovanou hypotézu – v našem případě hypotézu, že testovaná sekvence je náhodná. Oproti té stojí *alternativní hypotéza* H_A, která nějakým způsobem popírá nulovou hypotézu – v našem případě je to hypotéza, že testovaná sekvence není náhodná. Po provedení každého testu musí být učiněno rozhodnutí, na jehož základě je nulová hypotéza *zamítnuta* či *nezamítnuta*.

Pro toto rozhodnutí použijeme tzv. *testovou statistiku*[1], jejíž obor hodnot rozdělíme na dva disjunktní obory – *obor zamítnutí (kritický obor)* a *obor přijetí*. Hodnoty testové statistiky, které oddělují obor zamítnutí od oboru přijetí se nazývají *kritické hodnoty*, a můžeme je určit z teoretického rozložení testové statistiky pomocí matematických metod. Během testu je na základě dat (testované sekvence) vypočítána hodnota testové statistiky. Ta je pak porovnána s kritickou hodnotou, a pokud padne do kritického oboru je nulová hypotéza zamítnuta (a je přijata alternativní hypotéza). V opačném případě je nulová hypotéza nezamítnuta.

Důvodem proč testování statistických hypotéz funguje je, že kritické hodnoty jsou závislé na teoretickém rozložení hodnot testové statistiky (a to za předpokladu

[1] Připomeňme, že *statistiku* definujeme jako náhodnou veličinu (resp. náhodný vektor), která vzniká transformací jednoho nebo více náhodných výběrů.

náhodnosti). Předpokládáme-li, že testovaná data jsou také náhodná, měla by hodnota vypočítané testové statistiky přesáhnout kritickou hodnotu jen s velmi malou pravděpodobností (např. $0,001$ %). Pokud kritickou hodnotu přeci jen přesáhne, je předpoklad náhodnosti testovaných dat přinejmenším zpochybněn.

Při testování statistických hypotéz se můžeme dopustit i nesprávných rozhodnutí (viz tab. 4.1). Buď zamítneme nulovou hypotézu H_0, ačkoliv je ve skutečnosti pravdivá (tzv. *chyba I. druhu*), nebo nepravdivou nulovou hypotézu nezamítneme (tzv. *chyba II. druhu*). Pravděpodobnost, že se dopustíme chyby I. druhu se nazývá

	Závěr	
Skutečná situace	H_0 **se nezamítá**	H_0 **se zamítá**
H_0 je pravdivá	správné rozhodnutí	chyba I. druhu
H_0 je nepravdivá	chyba II. druhu	správné rozhodnutí

Tab. 4.1: Výsledky testu hypotézy.

hladina významnosti testu hypotézy. Typicky bývá nastavena před začátkem testu a označujeme ji α. V našem případě to je pravděpodobnost, že výsledek testu náhodné sekvence bude značit, že je nenáhodná. To odpovídá situaci, kdy má testovaná sekvence nenáhodné vlastnosti, ačkoli byla vygenerována dobrým PRNG. Běžná hodnota α se u statistických testů v kryptografii pohybuje mezi[2] $0,01$ a $0,0001$. Pravděpodobnost, že se dopustíme chyby II. druhu označujeme β. V našem případě to je pravděpodobnost, že výsledek testu nenáhodné sekvence bude značit, že je náhodná. To odpovídá situaci, kdy je kvalitní náhodná sekvence vygenerována špatným PRNG.

V ideálním případě samozřejmě požadujeme, aby pravděpodobnosti obou těchto chyb byly co možná nejmenší. Při pevném rozsahu výběrového souboru ovšem platí, že čím menší je pravděpodobnost chyby I. druhu, tím větší je pravděpodobnost chyby II. druhu a naopak. Celý testovací postup je proto odvozen tak, aby při předem dané hladině významnosti α zajišťoval minimální pravděpodobnost výskytu chyby II. druhu. Tím je vlastně minimalizována pravděpodobnost nezamítnutí kvalitní náhodné sekvence vygenerované špatným generátorem. Test hypotézy použitím klasického přístupu vypadá následovně:

1. Formulujeme nulovou a alternativní hypotézu.
2. Zvolíme hladinu významnosti α.
3. Určíme kritickou hodnotu (resp. kritické hodnoty).
4. Vypočítáme hodnotu testové statistiky.
5. Pokud hodnota testové statistiky padne do kritického oboru, tak H_0 zamítáme. V opačném případě H_0 nezamítáme.

Jak vidíme, je u tohoto přístupu k testování hypotéz hladina významnosti stanovena předem a závěrem je pak zamítnutí či nezamítnutí nulové hypotézy. To však nedovoluje uživatelům, kteří mají mají k dispozici pouze závěry o testované hypotéze,

[2]Hladinu významnosti $\alpha = 0,0001$ definuje například FIPS 140-2 při testech prováděných po spuštění či restartu kryptografického modulu.

vybrat svoji vlastní hladinu významnosti a učinit tak své vlastní ohodnocení. To je důvodem pro zavedení tzv. *P-hodnoty* hypotézy, což je přesně nejmenší pozorovaná hladina významnosti, na které nulová hypotéza může být zamítnuta. Jinými slovy je to pravděpodobnost, že perfektní PRNG vygeneruje sekvenci stejně či méně náhodnou, než je testovaná sekvence. Je-li P-hodnota rovna 1, jeví se testovaná sekvence jako perfektně náhodná. Naopak P-hodnota rovna 0 značí, že je sekvence naprosto nenáhodná. Test hypotézy založený na použití P-hodnoty pak vypadá následovně:

1. Formulujeme nulovou a alternativní hypotézu.
2. Zvolíme hladinu významnosti α.
3. Vypočítáme hodnotu testové statistiky.
4. Určíme (vypočítáme) P-hodnotu.
5. Pokud $P \leq \alpha$, tak H_0 zamítáme[3]. V opačném případě H_0 nezamítáme.

Je-li například $\alpha = 0,01$ můžeme očekávat, že 1 sekvence ze 100 bude zamítnuta. $P > 0,01$ bude znamenat, že sekvence bude považována za náhodnou s jistotou na 99 %. $P \leq 0,01$ bude znamenat, že sekvence bude považována za nenáhodnou s jistotou na 99 %.

4.3 Praktická aplikace testů

K provedení testů náhodnosti jsme se rozhodli použít baterii testů od NIST (verze programu 1.8 z března 2005) [FIP00]. Tato implementace testů NIST je veřejnosti dostupná včetně zdrojového kódu (je naprogramována v MS Visual C++) a není předmětem žádných ochran autorských práv. Celkově tato baterie obsahuje 16 statistických testů:

1. The Frequency (Monobit) Test;
2. Frequency Test within a Block;
3. The Runs Test;
4. Test for the Longest-Run-of-Ones in a Block;
5. The Binary Matrix Rank Test;
6. The Discrete Fourier Transform (Spectral) Test;
7. The Non-overlapping Template Matching Test;
8. The Overlapping Template Matching Test;
9. Maurer's Universal Statistical Test;
10. The Lempel-Ziv Compression Test;

[3]Oproti běžné literatuře stačí podle NIST pokud je $P < \alpha$.

11. The Linear Complexity Test;
12. The Serial Test;
13. The Approximate Entropy Test;
14. The Cumulative Sums (Cusums) Test;
15. The Random Excursions Test;
16. The Random Excursions Variant Test.

Oproti dokumentaci, však nejnovější verze programu neobsahuje Lempel-Ziv kompresní test a jsou také upraveny některé hodnoty spektrálního testu.

Při aplikaci těchto testů a interpretaci výsledků se často nějakým způsobem odkazujeme na teoretické standardizované normální rozložení ($N(0,1)$) a Pearsonovo chí-kvadrát rozložení (χ^2). Testová statistika pro standardizované normální rozložení je $z = (x - \mu)/\sigma$, kde μ je střední hodnota a σ je směrodatná odchylka. Testová statistika pro chí-kvadrát rozložení je $\chi^2 = \Sigma((o_i - e_i)^2/e_i)$, kde o_i jsou pozorované empirické četnosti a e_i očekávané empirické četnosti.

Strategie testování generátorů náhodných čísel a interpretace výsledků je následující:

1. Zvolíme generátor – v našem případě TRNG umístěný na jedné ze čtyř vybraných čipových karet.
2. Vygenerujeme vhodný počet náhodných sekvencí – při testování TRNG nám stačí pouze jedna dlouhá sekvence (v našem případě o délce 50 MB).
3. Aplikujeme vybrané statistické testy – ty ohodnotí danou sekvenci a vrátí jednu (či více) P-hodnot. Každý test aplikovatelný na danou sekvenci, může být aplikovatelný i na její podsekvence. Je-li sekvence náhodná, pak každá podsekvence by měla být také náhodná.
4. Prozkoumáme získané P-hodnoty a zhodnotíme počet zamítnutí/nezamítnutí nulové hypotézy – pro fixní hladinu významnosti se očekává určité procento zamítnutí.

Interpretace výsledků testování může být vedena několika způsoby. NIST doporučuje jak prověření počtu (resp. podílu či procenta) sekvencí, které prošly testem, tak také kontrolu rovnoměrnosti rozložení výsledných P-hodnot.

Testujeme-li například $m = 1000$ náhodných sekvencí na hladině významnosti $\alpha = 0,01$ a 996 z nich má $P \geq 0,01$, tak jejich podíl je $996/1000 = 0,996$. Rozsah přijetí je určen intervalem spolehlivosti jako $(1 - \alpha) \pm 3 \cdot \sqrt{((1 - \alpha)\alpha)/m}$, což po dosazení dává $0,99 \pm 3 \cdot \sqrt{(0,99 \cdot 0,01)/1000} = 0,99 \pm 0,0094392$. Výsledné hodnota by tedy měla ležet nad[4] $0,9805607$.

Kontrola rovnoměrnosti rozložení P-hodnot je provedena pomocí χ^2 testu dobré shody[5]. Interval mezi 0 a 1 se rozdělí na 10 stejně velkých podintervalů a vypočítá

[4] Jedná se nám o vymezení spodní hranice.

[5] Obecně lze ale použít také jiné testy (např. Kolmogorov-Smirnov test dobré shody).

se počet P-hodnot v každém z nich (čili vizuálně se v podstatě jedná o vytvoření histogramu). Testová statistika je $\chi^2 = \Sigma_{i=1}^{10}((o_i - e_i)^2/e_i)$, kde pozorované empirické četnosti o_i udává počet P-hodnot v podintervalu i a očekávané empirické četnosti e_i vypočítáme jako $s \cdot p_i = s/10$ (tj. počet testovaných sekvencí krát pravděpodobnost jejich výskytu v daném podintervalu). Pokud je výsledná P-hodnota $P_T \geq 0,0001$, mohou být testované sekvence považovány za rovnoměrně rozložené.

Jestliže kterýkoliv z výše uvedených postupů testování selže (tj. odpovídající nulová hypotéza je zamítnuta), měly by být provedeny další statistické testy s nově vygenerovanými náhodnými sekvencemi. Na jejich základě se pak určí, zdali šlo o ojedinělou statistickou anomálii, nebo zda se skutečně jedná o jasný důkaz nenáhodnosti. Při vyhodnocování výsledků testů mohou ale obecně nastat tři případy: analýza výsledků bude jednoznačně značit náhodnost testovaných sekvencí; analýza výsledků bude jednoznačně značit nenáhodnost testovaných sekvencí; výsledky testů budou nejednoznačné (resp. neprůkazné).

4.3.1 Použité statistické testy

V této části práce si stručně popíšeme význam jednotlivých statistických testů. Jejich podrobný popis (včetně mnoha příkladů a odvození testů) lze nalézt v [FIP00]. Poznamenejme také, že mnohé z testů mají poměrně velké požadavky na délku vstupních sekvencí. To je způsobeno tím, že při výpočtu využívají k urychlení různých asymptotických aproximací, které vedou k dostatečně přesným výsledkům pouze pro dlouhé sekvence.

Frequency (Monobit) Test

Někdy je označován jako jednobitový POKER či Golomb 1 a je zaměřen na počet jedniček a nul vyskytujících se v celé testované sekvenci. Jeho účelem je zjištění počtu výskytu jedniček a nul, který by měl být pro skutečně náhodnou sekvenci přibližně stejný. Veškeré následující testy předpokládají, úspěšné absolvování tohoto testu. Délka testované sekvence musí být alespoň 100 bitů.

Frequency Test within a Block

Tento test je zaměřen na počet jedniček a nul vyskytujících se v M-bitovém bloku testované sekvence. Jeho účelem je zjištění počtu výskytu jedniček a nul v M-bitovém bloku, který by měl být pro skutečně náhodnou sekvenci přibližně $M/2$. Neúspěch v tomto testu značí velkou odchylku počtu jedniček (a tedy i nul) alespoň v jednom z bloků. Velikost bloku by měla být $M \geq 20$, přičemž pro $M = 1$ tento test degeneruje na předchozí jednobitový POKER test. Délka testované sekvence musí být opět alespoň 100 bitů.

The Runs Test

Někdy je označován jako Golomb 2 a je zaměřen na zjištění celkového počtu běhů (tj. nepřerušených posloupností identických bitů) v testované sekvenci. Běh délky k je právě k identických bitů zepředu i zezadu ohraničených bity opačné hodnoty.

Účelem testu je určit zdali počet běhů (různých délek) jedniček a nul odpovídá skutečně náhodné sekvenci. Jedná se tedy vlastně o určení, zdali je oscilace mezi nulami a jedničkami příliš rychlá či příliš pomalá. Délka testované sekvence musí být i u tohoto testu alespoň 100 bitů.

Test for the Longest-Run-of-Ones in a Block

Test se zaměřuje na zjištění nejdelšího běhu jedniček vyskytujících se v M-bitovém bloku testované sekvence. Délka tohoto běhu by měla být odpovídající délce nejdelšího běhu jedniček ve skutečně náhodné sekvenci. Nepravidelnost v očekávané délce nejdelšího běhu jedniček samozřejmě také implikuje nepravidelnost v očekávané délce nejdelšího běhu nul, a proto je samostatný test jedniček zcela dostačující. Délka testované sekvence závisí na velikosti použitého bloku: pro $M = 8$ je minimální délka 128 bitů; pro $M = 128$ je minimální délka 6 272 bitů; a pro $M = 10\,000$ je minimální délka 750 000 bitů. Změna nastavení velikosti bloku však vyžaduje zásah do zdrojového kódu a opětovnou kompilaci programu.

The Binary Matrix Rank Test

Jiný přístup k testování náhodnosti je kontrola lineární závislosti mezi stejně dlouhými podřetězci celé testované sekvence. Tento test (který je převzat z baterie DIEHARD) funguje tak, že z testované sekvence konstruuje matice po sobě jdoucích jedniček a nul, a kontroluje lineární závislosti mezi řádky či sloupci zkonstruovaných matic. Protože pro test je potřeba vytvořit alespoň 38 matic o velikosti 32×32, musí být testovaná sekvence dlouhá alespoň 38 912 bitů.

The Discrete Fourier Transform (Spectral) Test

Test je založen na diskrétní Fourierově transformaci. Je zaměřen na detekci periodických vlastností (tj. opakujících se vzorů) testované sekvence. Délka testované sekvence musí být alespoň 1 000 bitů.

The Non-overlapping Template Matching Test

Tento test je zaměřen na počet výskytů předdefinovaných řetězců (resp. šablon) v testované sekvenci. Jeho účelem je odhalení příliš mnoho či naopak příliš málo výskytů předdefinovaných (neperiodických) m-bitových šablon. Společně s následujícím testem, používá k vyhledávání m-bitové šablony m-bitové okno. Není-li šablona nalezena, okno se posune o jednu pozici dopředu a pokračuje se ve srovnávání a vyhledávání. Pokud šablona je nalezena, přesune se okno na prví bit za šablonou a opět se pokračuje ve srovnávání a vyhledávání. Testovací program podporuje velikosti šablon $m = 2$ až $m = 10$, ale pro dosažení smysluplnějších výsledků je doporučeno používání hodnot $m = 9$ či $m = 10$. Délka testované sekvence musí být alespoň 1 000 000 bitů.

The Overlapping Template Matching Test

Stejně jako v předchozím případě je tento test zaměřen na počet výskytů předdefinovaných řetězců (resp. šablon) v testované sekvenci. K vyhledávání m-bitové šablony využívá také m-bitové okno. Není-li šablona nalezena, okno se posune o jednu pozici dopředu a pokračuje se ve srovnávání a vyhledávání. Rozdíl oproti předcházejícímu testu je v tom, že pokud šablona je nalezena, posune se okno také jen o jednu pozici dopředu a opět se pokračuje ve srovnávání a vyhledávání. Testovací program podporuje různé velikosti šablon, ale pro dosažení smysluplnějších výsledků je doporučeno používání hodnot $m = 9$ či $m = 10$. Délka testované sekvence musí být alespoň 1 000 000 bitů.

Maurer's Universal Statistical Test

Tento test není navržen k detekci jednoho konkrétního statistického nedostatku. Jeho účelem je detekce, zdali může být testovaná sekvence výrazně bez ztráty informace zkomprimována. Snadno zkomprimovatelná sekvence je pak považována za nenáhodnou. Vstupy testu jsou délka bloku (L), počet bloků v inicializační sekvenci (Q), a testovaná sekvence délky n bitů. Hodnoty L a Q by měly být v závislosti na n zvoleny podle tabulky 4.2.

n	L	Q
$\geq$ 387 840	6	640
$\geq$ 904 960	7	1 280
$\geq$ 2 068 480	8	2 560
$\geq$ 4 654 080	9	5 120
$\geq$ 10 342 400	10	10 240
$\geq$ 22 753 280	11	20 480
$\geq$ 49 643 520	12	40 960
$\geq$ 107 560 960	13	81 920
$\geq$ 231 669 760	14	163 840
$\geq$ 496 435 200	15	327 680
$\geq$ 1 059 061 760	16	655 360

Tab. 4.2: Hodnoty vstupních parametrů testu.

The Linear Complexity Test

Tento test se zaměřuje na délku posuvného registru s lineární zpětnou vazbou (LFSR – linear feedback shift register). Jeho účelem je zjistit, zdali je či není testovaná sekvence natolik složitá, aby mohla být považována za náhodnou. Skutečně náhodné sekvence se vyznačují dlouhými LFSR, zatímco krátké LFSR bývají znakem nenáhodnosti. Vstupem testu je velikost bloku $500 \leq M \leq 5000$ a testovaná sekvence o délce alespoň 1 000 000 bitů.

The Serial Test

Test se zaměřuje na počet všech možných výskytů překrývajících se m-bitových vzorků. Jeho účelem je zjistit, zdali je počet výskytů 2^m překrývajících se m-bitových vzorků přibližně stejný, jak by se očekávalo u skutečně náhodné sekvence. Celkově se skládá ze dvou testových statistik, a jeho výsledkem jsou tedy také dvě P-hodnoty. Pro $m = 1$ tento test degeneruje na jednobitový POKER test. Je-li n počet bitů testované sekvence, je třeba volit $m < \lfloor log_2 n \rfloor - 2$.

The Approximate Entropy Test

Stejně jako předcházející test, zaměřuje se i tento na počet všech možných výskytů překrývajících se m-bitových vzorků. Jeho účelem je srovnání výskytů překrývajících se bloků sousedních délek (m a $m + 1$ bitů) oproti jejich očekávanému výskytu ve skutečně náhodné sekvenci. Je-li n počet bitů testované sekvence, je třeba volit $m < \lfloor log_2 n \rfloor - 2$.

The Cumulative Sums (Cusums) Test

Tento test je zaměřen na maximální vychýlení (od nuly) náhodné „vycházky“, definované kumulativním součtem upravených číslic (reprezentovaných hodnotami $-1, +1$) v testované sekvenci. Jeho účelem je určit, zdali kumulativní součet dílčích sekvencí je příliš velký či příliš malý oproti očekávanému kumulativnímu součtu skutečně náhodné sekvence (ten by měl být blízký nule). Malé hodnoty kumulativního součtu značí, že nuly a jedničky jsou zamíchány rovnoměrně. Testuje se náhodná vycházka od začátku i od konce sekvence (výsledkem jsou tedy dvě P-hodnoty). Délka testované sekvence musí být alespoň 100 bitů.

The Random Excursions Test

Test je zaměřen na počet cyklů, které mají právě K návštěv v náhodné „vycházce“, definované opět kumulativním součtem upravených číslic. Účelem testu je určit, zdali se počet návštěv jednotlivých stavů (resp. hodnot) uvnitř cyklu liší oproti očekávanému počtu u skutečně náhodné sekvence. Jedná se vlastně o sérii osmi testů (a také závěrů) pro stavy $-4, -3, -2, -1, +1, +2, +3, +4$. Délka testované sekvence musí být alespoň 1 000 000 bitů.

The Random Excursions Variant Test

Tento test se zaměřuje na celkový počet návštěv jednotlivých stavů (resp. hodnot) v náhodné „vycházce“, definované opět kumulativním součtem upravených číslic. Účelem testu je určit, zdali se počet návštěv jednotlivých stavů u testované sekvence liší oproti očekávanému počtu u skutečně náhodné sekvence. Jedná se vlastně o sérii šestnácti testů (a také závěrů) pro stavy $-9, -8, \ldots, -1, +1, \ldots, +8, +9$. Délka testované sekvence musí být opět alespoň 1 000 000 bitů.

4.3.2 Výsledky testování

V této části práce shrneme závěrečné výsledky provedených statistických testů. Podrobné výpisy, včetně všech detailů týkajících se nastavení testů, jsou dosti rozsáhlé a jsou k dispozici na přiloženém CD.

Sekvence o velikosti 1GB

Nejdříve jsme testovali náhodnou sekvenci o velikosti 1GB, která byla vygenerována čipovou kartou GXPLite-Generic. Hladinu významnosti jsme zvolili $\alpha = 0,01$ a v dalším testu[6] $\alpha = 0,001$. V případě $\alpha = 0,01$ byla sekvence rozdělena na 100 dlouhých podsekvencí, zatímco v případě $\alpha = 0,001$ byla sekvence rozdělena na 1000 kratších podsekvencí. Aby sekvence byla dělitelná nejprve 100 a později 1000, byla její velikost postupně redukována. Bylo ponecháno standardní (doporučené) nastavení testů. Z parametrů závisejících na velikosti testovaných podsekvencí jsme přizpůsobovali pouze hodnoty L a Q u Maurerova univerzálního testu.

Celkově dopadla většina provedených testů úspěšně a jejich výsledky jsou (s výjimkou[7] testů The Non-overlapping Template Matching, The Random Excursions, The Random Excursions Variant) shrnuty do tabulky 4.3. Připomeňme, že podle

	$\alpha = 0,01$		$\alpha = 0,001$	
Název testu	**P-hodnota**	**Podíl**	**P-hodnota**	**Podíl**
Frequency	0,554420	1,0000	0,614226	0,9990
Block Frequency	0,334538	1,0000	0,082513	0,9990
Runs	0,739918	0,9900	0,504219	0,9990
Longest Run	0,834308	1,0000	0,807412	1,0000
Binary Matrix Rank	0,383827	1,0000	0,049030	1,0000
Discrete Fourier Transform	0,455937	0,9900	0,424453	0,9960
Overlapping Templates	0,045675	0,9800	0,241741	0,9980
Universal	0,779188	0,9900	0,655854	0,9990
Linear Complexity	0,162606	1,0000	0,781106	0,9990
Serial (Test Statistic 1)	0,401199	1,0000	0,570792	0,9980
Serial (Test Statistic 2)	0,002374	1,0000	0,130369	0,9970
Approximate Entropy	0,834308	0,9600	0,415422	1,0000
Cumulative Sums (Forward)	0,657933	0,9900	0,616305	0,9990
Cumulative Sums (Reverse)	0,834308	1,0000	0,442831	0,9970

Tab. 4.3: GXPLite-Generic – sekvence o velikosti 1GB.

NIST by pro zamítnutí H_0 na základě výsledné P-hodnoty muselo platit, že $P_T < 0,0001$. Pro 100 sekvencí na hladině významnosti $\alpha = 0,01$ byla dolní hranice počtu (resp. podílu či procenta) sekvencí, které prošly testem, stanovena na $0,960150$. Pro 1000 sekvencí na hladině významnosti $\alpha = 0,001$ byla tato dolní hranice stanovena na $0,980518$. Výsledné hodnoty by tedy v případě skutečně náhodných sekvencí měly ležet nad těmito hranicemi.

[6] Změna hladiny významnosti na jinou než přednastavenou hodnotu $\alpha = 0,01$ vyžaduje zásah do zdrojového kódu a opětovnou kompilaci programu.

[7] Výsledky těchto testů jsou poměrně rozsáhlé a jsou také k dispozici na přiloženém CD.

Jeden z testů The Non-overlapping Template Matching dopadl na hladině významnosti $\alpha = 0,01$ neúspěšně (podíl nezamítnutých sekvencí byl jen $0,9500$). Je však vidět, že odchylka od $0,960150$ způsobená jednou chybnou sekvencí navíc je jen nepatrná. Pro hladinu významnosti $\alpha = 0,001$ dopadly všechny statistické testy úspěšně.

Sekvence o velikosti 50MB

Dále jsme testovali náhodné sekvence o velikosti 50MB, které jsme měli vygenerované všemi kartami. Veškeré testování probíhalo na hladině významnosti $\alpha = 0,01$. Na základě osobní e-mailové komunikace s Luďkem Smolíkem [Smo05] jsme se dozvěděli, že je dobré testovat i samotný počet rozdělení na podsekvence.

Každou sekvenci jsme proto rozdělili nejprve na 100, pak na 400 a na závěr na 800 podsekvencí. Více sekvencí na stejné hladině významnosti přispělo ke zvýšení přesnosti měření. Bylo ponecháno opět standardní (doporučené) nastavení testů. Z parametrů závisejících na velikosti testovaných podsekvencí jsme přizpůsobovali opět pouze hodnoty L a Q u Maurerova univerzálního testu. Na všechny tyto podsekvence bylo aplikováno celkem 9 statistických testů[8]. Všechny provedené testy dopadly úspěšně.

Výsledky testů aplikovaných na 800 podsekvencí[9] jsou shrnuty do tabulek 4.4 a 4.5. Tabulka 4.4 obsahuje výsledky statistických testů sekvencí vygenerovaných

	GXP211 PK		GXP211 PK IS	
Název testu	**P-hodnota**	**Podíl**	**P-hodnota**	**Podíl**
Frequency	0,642325	0,9875	0,531629	0,9875
Block Frequency	0,063815	0,9912	0,033197	0,9925
Runs	0,853464	0,9838	0,707002	0,9850
Binary Matrix Rank	0,932904	0,9912	0,247698	0,9925
Discrete Fourier Transform	0,033751	0,9862	0,441902	0,9912
Universal	0,036054	0,9838	0,385973	0,9825
Serial (Test Statistic 1)	0,701879	0,9875	0,181557	0,9862
Serial (Test Statistic 2)	0,873597	0,9900	0,175450	0,9900
Approximate Entropy	0,167184	0,9938	0,224821	0,9938
Cumulative Sums (Forward)	0,403403	0,9912	0,369006	0,9875
Cumulative Sums (Reverse)	0,897763	0,9888	0,487074	0,9812

Tab. 4.4: GXP211 PK, GXP211 PK IS – sekvence o velikosti 50MB.

čipovými kartami GXP211 PK a GXP211 PK IS. Označení IS (international sample) u druhé karty značí, že se jedná o mezinárodní verzi splňující Francouzské státní restrikce při používání kryptografie. Tabulka 4.5 obsahuje výsledky statistických testů sekvencí vygenerovaných čipovými kartami GXPLite-Generic a GXPPro R3.

[8] Po rozdělení na 800 podsekvencí byla délka jedné podsekvence 524 288 bitů, což znemožnilo aplikaci zbylých šesti statistických testů.

[9] Výsledky testů pro rozdělení 50MB sekvence na 400 a na 100 podsekvencí jsou k dispozici na přiloženém CD.

Pro 800 sekvencí na hladině významnosti $\alpha = 0,01$ byla dolní hranice počtu (resp. podílu či procenta) sekvencí, které prošly testem, stanovena na $0,979447$.

	GXPLite-Generic		GXPPro R3	
Název testu	**P-hodnota**	**Podíl**	**P-hodnota**	**Podíl**
Frequency	0,938463	0,9900	0,942460	0,9900
Block Frequency	0,428095	0,9888	0,757297	0,9938
Runs	0,242986	0,9912	0,230755	0,9912
Binary Matrix Rank	0,235285	0,9950	0,455937	0,9800
Discrete Fourier Transform	0,319084	0,9875	0,018223	0,9900
Universal	0,084938	0,9875	0,118812	0,9900
Serial (Test Statistic 1)	0,169512	0,9900	0,832124	0,9862
Serial (Test Statistic 2)	0,919445	0,9850	0,681272	0,9912
Approximate Entropy	0,567201	0,9838	0,300464	0,9912
Cumulative Sums (Forward)	0,249284	0,9925	0,174249	0,9912
Cumulative Sums (Reverse)	0,373203	0,9900	0,158133	0,9925

Tab. 4.5: GXPLite-Generic, GXPPro R3 – sekvence o velikosti 50MB.

Kapitola 5

Závěr

V druhé kapitole jsme se seznámili se základními kryptografickými eskalačními protokoly. Viděli jsme, že se jedná o mimořádnou třídu metod, které jsou založeny na použití dat s nízkou entropií (jako například hesel) a zajišťují autentizované ustavení klíčů přes nezabezpečený kanál. Při vlastním popisu eskalačních protokolů jsme se zaměřili jak na principy jejich návrhu, tak také na bezpečnostní problémy, které během jejich vytváření vznikají a musí být nějakým způsobem vyřešeny. V současné době se standardizaci kryptografických eskalačních protokolů věnuje pracovní skupina IEEE P1363.

Kvalita výsledných kryptografických klíčů však závisí především na generátorech náhodných či pseudonáhodných sekvencí, které jednotlivé strany v průběhu protokolu používají. Těmito generátory jsme se zabývali v kapitole třetí, kde jsme se zaměřili především na generátory vhodné pro kryptografické čipové karty. Seznámili jsme se jak s problematikou generování skutečně náhodných sekvencí, tak také s problematikou generování pseudonáhodných sekvencí. Naší pozornosti neunikly ani kryptoanalytické útoky na generátory pseudonáhodných sekvencí. Během práce na této kapitole byla zjištěna chyba v popisu algoritmu ANSI X9.17 PRNG publikovaného v [MvOV01].

Praktickou částí práce byl test generátorů skutečně náhodných sekvencí implementovaných na vybraných kryptografických čipových kartách. K statistickému testování (které ale v žádném případě nenahrazuje kryptoanalýzu) byla použita baterie testů NIST. Velmi cenné rady a informace o metodice testování jsme získali také na základě osobní e-mailové komunikace s Luďkem Smolíkem [Smo05]. Testy všech generátorů implementovaných na vybraných kryptografických čipových kartách dopadly za standardních vnějších podmínek úspěšně. Podrobný popis průběhu testování a vyhodnocení testů je obsažen v kapitole čtvrté.

Otevřenou otázkou zůstává, zdali by provedené statistické testy dopadly stejně úspěšně i za nestandardních vnějších podmínek (např. změny napětí a proudu, hodinového taktu, teploty či radiace). Tomuto problému bych se chtěl věnovat ve svém dalším výzkumu.

Reference

[ANS85] ANSI X9.17 (Revised), American National Standard for Financial Institution Key Management (Wholesale), American Bankers Association, May 1985.

[ARV95] W. Aiello, S. Rajagopalan, and R. Venkatesan. Design of Practical and Provably Good Random Number Generators. In *5th Annual ACM-SIAM Symposium of Discrete Algorithms*, pages 1–8, 1995.

[ARV99] W. Aiello, S. Rajagopalan, and R. Venkatesan. High-Speed Pseudorandom Number Generation with Small Memory. volume 1636 of *Lecture Notes in Computer Science*, pages 290–304. Springer, 1999.

[BB99] V. Bagini and M. Bucci. A Design of Reliable True Random Number Generator for Cryptographic Applications. In *Proceedings of the 1st Workshop Cryptographic Hardware and Embedded Systems (CHES) 1999*, volume 1717 of *Lecture Notes in Computer Science*, pages 204–218. Springer, 1999.

[BBL04] H. Bock, M. Bucci, and R. Luzzi. An Offset-Compensated Oscillator-Based Random Bit Source for Security Applications. In *Cryptographic Hardware and Embedded Systems (CHES) 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 268–281. Springer, 2004.

[BGL+03] M. Bucci, L. Germani, R. Luzzi, A. Trifiletti, and M. Varanonuovo. A High-Speed Oscillator-Based Truly Random Number Source for Cryptographic Applications on a Smart Card IC. In *IEEE Transactions Computers*, volume 52, pages 403–409, April 2003. Available at: `http://csdl.computer.org/dl/trans/tc/2003/04/t0403.pdf`.

[BM92] S. M. Bellovin and M. Merritt. Encrypted Key Exchange: Password-based protocols secure against dictionary attacks. In *Proceedings IEEE Computer Society symposium on Research in Security and Privacy*, pages 72–84, May 1992. Available at: `http://www1.cs.columbia.edu/~smb/papers/neke.pdf`.

[BM93] S. M. Bellovin and M. Merritt. Augmented Encrypted Key Exchange: a Password-Based Protocol Secure Against Dictionary Attacks and Password File Compromise. In *Proceedings of the First ACM Conference*

on Computer and Communications Security, pages 244–250, November 1993. Available at: `http://www1.cs.columbia.edu/~smb/papers/aeke.pdf`.

[BM94] S. M. Bellovin and M. Merritt. An attack on the Interlock Protocol When Used for Authentication. In *IEEE Transactions on Information Theory*, volume 40, pages 273–275, January 1994. Available at: `http://www1.cs.columbia.edu/~smb/papers/interlock.pdf`.

[BMO01] M. Budíková, Š. Mikoláš, and P. Osecký. *Teorie pravděpodobnosti a matematická statistika*. Masaryk University in Brno, 2001. ISBN 80-210-1832-1.

[Boy89] J. Boyar. Inferring sequences produced by pseudo-random number generators. In *Journal of the ACM (JACM)*, volume 36, pages 129–141, 1989.

[BR00] M. Bellare and P. Rogaway. The AuthA Protocol for Password-based Authenticated KeyExchange. In *Contribution to the IEEE P1363 study group*, February 2000. Available at: `http://grouper.ieee.org/groups/1363/passwdPK/contributions/autha.pdf`.

[BTSL01] M. Bucci, E. Trichina, D. De Seta, and R. Luzzi. Supplemental Cryptographic Hardware for Smart Cards. In *IEEE Micro*, volume 21, pages 26–35, December 2001. Available at: `http://csdl.computer.org/dl/mags/mi/2001/06/m6026.pdf`.

[Cry] Information Security Research Centre at Queensland. Crypt-XS Test Suite, University of Technology in Australia. Available at: `http://www.isrc.qut.edu.au/resource/cryptx/index.php`.

[Dai97] W. Dai. Crypto++ library, 1997. Available at: `http://www.eskimo.com/~weidai/cryptlib.html`.

[DH76] W. Diffie and M. E. Hellman. New Directions in Cryptography. In *IEEE Transactions on Information Theory*, volume IT-22, pages 644–654, November 1976.

[DHY02] A. Desai, A. Hevia, and Y. L. Yin. A Practice-Oriented Treatment of Pseudorandom Number Generators. In *EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 368–383. Springer, 2002.

[Dic03] M. Dichtl. How to Predict the Output of a Hardware Random Number Generator. In *Cryptographic Hardware and Embedded Systems (CHES) 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 181–188. Springer, 2003. Available at: `http://eprint.iacr.org/2003/051.ps`.

[DP89] D. W. Davies and W. L. Price. *Security for computer networks*. John Wiley & Sons, Inc., second edition, 1989.

[DRV02] N. Dedic, L. Reyzin, and S. Vadhan. An Improved Pseudorandom Generator Based on Hardness of Factoring. In *Third Conference on Seciruty in Communication Networks (SCN '02)*, 2002. Available at: `http://eprint.iacr.org/2002/131.pdf`.

[DvOW92] W. Diffie, P. C. van Oorschot, and M. J. Wiener. Authentication and Authenticated Key Exchange. In *Designs, Codes and Cryptography*, pages 107–125, 1992.

[EHK+03] M. Epstein, L. Hars, R. Krasinski, M. Rosner, and H. Zheng. Design and Implementation of a True Random Number Generator Based on Digital Circuit Artifacts. In *Proceedings of the 5th Workshop Cryptographic Hardware and Embedded Systems (CHES) 2003*, volume 2279 of *Lecture Notes in Computer Science*, pages 152–165. Springer, 2003.

[FCRV03] A. Fort, F. Cortigiani, S. Rocchi, and V. Vignoli. Very High-Speed True Random Noise Generator. In *Analog Integrated Circuits and Signal Processing*, volume 34, pages 97–105, 2003.

[FIP94a] Federal Information Processing Standards Publication 140-1, Security Requirements for Cryptographic Modules, National Institute of Standards and Technology, January 1994. Available at: `http://csrc.nist.gov/publications/fips/fips140-1/fips1401.pdf`.

[FIP94b] Federal Information Processing Standards Publication 186, DIGITAL SIGNATURE STANDARD (DSS), National Institute of Standards and Technology, May 1994. Available at: `http://www.itl.nist.gov/fipspubs/fip186.htm`.

[FIP00] Federal Information Processing Standards Special Publication 800-22. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, National Institute of Standards and Technology, October 2000. Available at: `http://csrc.nist.gov/publications/nistpubs/800-22/sp-800-22-051501.pdf`. Revised May 2001.

[FIP01] Federal Information Processing Standards Publication 140-2, Security Requirements for Cryptographic Modules, National Institute of Standards and Technology, May 2001. Available at: `http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf`.

[FIP02] Federal Information Processing Standards Communications Security Establishment, Implementation Guidance for FIPS PUB 140-1 and the Cryptographic Module Validation Program, January 2002. Available at: `http://csrc.nist.gov/cryptval/140-1/FIPS1401IG.pdf`.

[FMC84] R. C. Fairfield, R. L. Mortenson, and K. B. Coulthart. An LSI Random Number Generator (RNG). In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 203–230, 1984. Available at: `http://dsns.csie.nctu.edu.tw/research/crypto/HTML/PDF/C84/203.PDF`.

[Gen00] R. Gennaro. An Improved Pseudo-Random Generator Based on the Discrete Logarithm Problem. In *Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, pages 469–481, 2000. Available at: `http://www.research.ibm.com/security/newprng.ps`.

[HCD97] W. T. Holman, J. A. Connelly, and A. B. Dowlatabadi. An Integrated Analog/Digital Random Noise Source. In *IEEE Transactions on Circuits and System I: Fundamental Theory and Applications*, volume 44, pages 204–218, June 1997.

[HSS04] J. C. Hernández, J. M. Sierra, and A. Seznec. The SAC Test: A New Randomness Test, with Some Applications to PRNG Analysis. In *ICCSA 2004*, volume 3043 of *Lecture Notes in Computer Science*, pages 960–967. Springer, 2004.

[IEE05] IEEE. P1363.2/D20 (Draft version 20) - Standard Specifications for Password-based Public Key Cryptographic Techniques, 2005.

[Jab96] D. Jablon. Strong password-only authenticated key exchange. In *Computer Communication Review*, volume 26, pages 5–26. ACM SIGCOMM, October 1996. Available at: `http://www.jablon.org/jab96.pdf`.

[Jab97] D. Jablon. Extended Password Key Exchange Protocols Immune to Dictionary Attacks. In *Proceedings of the Sixth Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET-ICE '97)*, pages 248–255. IEEE Computer Society, June 1997. Available at: `http://www.jablon.org/jab97.pdf`.

[Jas96] B. Jaspan. Dual-workfactor Encrypted Key Exchange: Efficiently Preventing Password Chaining and Dictionary Attacks. In *Proceedings of the sixth USENIX UNIX Security Symposium*, pages 43–50, July 1996. Available at: `http://www.usenix.org/publications/library/proceedings/sec96/full_papers/jaspan/jaspan.ps`.

[JK99] B. Jun and P. Kocher. The Intel Random Number Generator. Cryptography Research, 1999. Available at: `http://www.cryptography.com/resources/whitepapers/IntelRNG.pdf`.

[Kat05] Katholieke Universiteit Leuven, Recent Collision Attacks on Hash Functions, European Network of Excellence in Cryptology (ECRYPT), February 2005. Available at: `http://www.ecrypt.eu.org/documents/STVL-ERICS-2-HASH_STMT-1.1.pdf`.

[KC02] D. J. Kinniment and E. G. Chester. Design of an On-Chip Random Number Generator using Metastability. In *Proceedings of the 28th European Solid-State Circuit Conference (ESSCIRC) 2002*, pages 595–598, September 2002. Available at: `http://www.staff.ncl.ac.uk/david.kinniment/Research/papers/ESSCIRC2002.PDF`.

[Kel05] Sharon S. Keller. NIST-Recommended Random Number Generator Based on ANSI X9.31 Appendix A.2.4 Using the 3-Key Triple DES and AES Algorithms, January 2005. Available at: `http://csrc.nist.gov/cryptval/rng/931rngext.pdf`.

[Knu97] D. E. Knuth. *The Art of Computer Programming: Seminumerical Algorithms, Volume 2*. Addison Wesley, third edition, 1997. ISBN 0-201-89684-2.

[Koe] F. Koeune. Pseudo-random number generator.

[Kra90] H. Krawczyk. How to predict congruential generators. In *Advances in Cryptology-CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 138–153. Springer, 1990. Available at: `http://dsns.csie.nctu.edu.tw/research/crypto/HTML/PDF/C89/138.PDF`.

[KSF99] J. Kelsey, B. Schneier, and N. Ferguson. Yarrow-160: Notes on the Design and Analysis of the Yarrow Cryptographic Pseudorandom Number Generator. In *Sixth Annual Workshop on Selected Areas in Cryptography*. Springer, August 1999. Available at: `http://www.secinf.net/uplarticle/4/yarrow-full.pdf`.

[KSWH98] J. Kelsey, B. Schneier, D. Wagner, and C. Hall. Cryptanalytic Attacks on Pseudorandom Number Generators. In *Fifth International Workshop Proceedings of Fast Software Encryption*, pages 168–188. Springer, March 1998. Available at: `http://www.schneier.com/paper-prngs.pdf`.

[Kwo00] T. Kwon. Ultimate Solution to Authentication via Memorable Password. In *Contribution to the IEEE P1363 study group for Future PKC Standards*, 2000. Available at: `http://grouper.ieee.org/groups/1363/passwdPK/contributions/amp.pdf`.

[Lab94] RSA Laboratories. RSAREF cryptographic library, March 1994. Available at: `ftp://ftp.funet.fi/pub/crypt/cryptography/asymmetric/rsa/rsaref2.tar.gz`.

[Len05] A. K. Lenstra. Further Progress in Hashing Cryptanalysis, February 2005. Available at: `http://cm.bell-labs.com/who/akl/hash.pdf`.

[LMS93] J. B. Lacy, D. P. Mitchell, and W. M. Schell. CryptoLib: Cryptography in Software. In *UNIX Security Symposium IV Proceedings*, pages 237–246. USENIX Association, 1993.

[Luc97] S. Lucks. Open key exchange: How to defeat dictionary attacks without encrypting public keys. In *Proceedings of the 5th International Workshop on Security Protocols*, volume 1361 of *LNCS*, pages 79–90. Springer, 1997. Available at: `http://grouper.ieee.org/groups/1363/passwdPK/contributions/amp.pdf`.

[Mac02] P. MacKenzie. The PAK suite: Protocols for Password-Authenticated Key Exchange. In *Contribution to the IEEE P1363 study group*, May 2002. Available at: `http://grouper.ieee.org/groups/1363/passwdPK/submissions/pak-suite.pdf`.

[Mar] G. Marsaglia, DIEHARD Statistical Tests. Available at: `http://stat.fsu.edu/pub/diehard/`.

[MSP00] P. MacKenzie, R. Swaminathan, and S. Patel. The PAK suite: Protocols for Password-Authenticated Key Exchange. In *Contribution to the IEEE P1363 study group*, August 2000. Available at: `http://cm.bell-labs.com/who/philmac/research/snapi-acfinal.pdf`.

[MT79] R. Morris and T. Thompson. Password security: a case history. In *Communications of the ACM*, volume 22, pages 594–597. ACM Press, November 1979.

[MT02] G. Marsaglia and W. W. Tang. Some difficult-to-pass tests of randomness. In *Journal of Statistical Software*, volume 7, 2002. Available at: `http://www.jstatsoft.org/v07/i03/tuftests.pdf`.

[MvOV01] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, August 2001. ISBN 0-8493-8523-7.

[Neu04] D. Neuenschwander. *Probabilistic and Statistical Methods in Cryptology*. Springer, 2004. ISBN 3-540-22001-1.

[Ng05] H. Ng. Simple Pseudorandom Number Generator with Strengthened Double Encryption (Cilia), 2005. Available at: `http://eprint.iacr.org/2005/086.pdf`.

[Nov99] J. Novovičová. *Pravděpodobnost a matematická statistika*. ČVUT, 1999.

[Pat97] S. Patel. Number theoretic attacks on secure password schemes. In *IEEE Symposium on Security and Privacy*, 1997. Available at: `http://csdl.computer.org/dl/proceedings/sp/1997/7828/00/78280236.pdf`.

[PC96] C. S. Petrie and J. A. Connelly. Modeling and Simulation of Oscillator-Based Random Number Generators. In *International Symposium on Circuits and Systems*, volume 6, pages 324–327, May 1996.

[PC00] C. S. Petrie and J. A. Connelly. A Noise-Based IC Random Number Generator for Applications in Cryptography. In *IEEE Transactions on Circuits and Systems*, volume 47, pages 615–621, May 2000.

[PH78] S. C. Pohlig and M. E. Hellman. An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. In *IEEE-IT*, volume IT-24, pages 106–110, 1978.

[PK99] R. J. Perlman and C. Kaufman. Secure Password-Based Protocol for Downloading a Private Key. In *Proceedings of the Internet Society Network and Distributed Systems Security Symposium*, 1999. Available at: `http://www.isoc.org/isoc/conferences/ndss/99/proceedings/papers/perlman.pdf`.

[PK01] R. Perlman and C. Kaufman. PDM: A New Strong Password-Based Protocol. In *Proceedings of the 10th USENIX Security Symposium*, pages 313–321, August 2001. Available at: `http://www.usenix.org/events/sec01/full_papers/kaufman/kaufman.pdf`.

[PKS00] M. G. Parker, A. H. Kemp, and S. J. Shepherd. Fast Blum-Blum-Shub Sequence Generation Using Montgomery Multiplication. In *IEEE Proceedings of Computers and Digital Techniques*, volume 147, pages 252–254, July 2000.

[RCW98] M. Roe, B. Christianson, and D. Wheeler. Secure Password-Based Protocol for Downloading a Private Key. Technical Report 445, University of Cambridge and University of Hertfordshire, July 1998. Available at: `http://research.microsoft.com/users/mroe/ccsr-tr4.pdf`.

[Ros00] J. Rosický. *Algebra – grupy a okruhy.* Masaryk University in Brno, 2000. ISBN 80-210-2303-1.

[RS84] R. L. Rivest and A. Shamir. How to Expose an Eavesdropper. In *Communications of the ACM*, volume 27, pages 393–395, 1984.

[Sch01] W. Schindler. Efficient Online Tests for True Random Number Generators. In *Proceedings of the Third International Workshop on Cryptographic Hardware and Embedded Systems*, pages 103–117, 2001.

[SJUH04] K. Song-Ju, K. Umeno, and A. Hasegawa. Corrections of the NIST Statistical Test Suite for Randomness, January 2004. Available at: `http://eprint.iacr.org/2004/018.pdf`.

[SK01] T. Stojanovski and L. Kocarev. Chaos-Based random number generators – Part I: Analysis. In *IEEE Transactions on Circuits System-1: Fundamental Theory and Applications*, volume 48, pages 281–288, 2001.

[Smo05] L. Smolík. Osobní e-mailová korespondence, 2005.

[Sot00] J. Soto. Statistical Testing of Random Number Generators. National Institute of Standards and Technology, 2000. Available at: `http://csrc.nist.gov/rng/nissc-paper.pdf`.

[STW95] M. Steiner, G. Tsudik, and M. Waidner. Refinement and Extension of Encrypted Key Exchange. In *Operating Systems Review*, volume 29, pages 22–30. ACM SIGOPS, July 1995.

[Sun99] Sun Microsystems. *Java Card 2.1 Application Programming Interface*, June 1999. Available at: `http://java.sun.com/products/javacard/JavaCard21API.pdf`.

[Tka02] T. E. Tkacik. A Hardware Random Number Generator. In *Cryptographic Hardware and Embedded Systems (CHES) 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 450–453. Springer, 2002.

[VIS05] VISA Industry Services, Visa Approved Chip Card Products, April 2005. Available at: `http://international.visa.com/fb/vendors/industryserv/approved.jsp`.

[vOW96] P. C. van Oorschot and M. J. Wiener. On Diffie-Hellman Key Agreement with Short Exponents. In *Advances in Cryptology – Eurocrypt 96*, volume 1070 of *Lecture Notes in Computer Science*, pages 332–343. Springer, 1996. Available at: `http://www3.sympatico.ca/wienerfamily/Michael/MichaelPapers/dhshortexp.pdf`.

[WF01] S. Walker and S. Foo. Evaluating Metastability in Electronic Circuits for Random Number Generation. In *Proceedings IEEE Computer Society Workshop on VLSI*, pages 99–101. IEEE Computer Society, April 2001. Available at: `http://csdl.computer.org/dl/proceedings/wvlsi/2001/1056/00/10560099.pdf`.

[Wu97] T. Wu. The Secure Remote Password protocol. In *Proceedings of the 1998 Internet Society Symposium on Network and Distributed Systems Security*, pages 97–111, November 1997. Available at: `ftp://srp.stanford.edu/pub/srp/srp.ps`.

[Wu02] T. Wu. SRP-6: Improvements and Refinements to the Secure Remote Password Protocoll. In *Submission to the IEEE P1363 Working Group*, October 2002. Available at: `http://srp.stanford.edu/srp6.ps`.

[Zen04] E. Zenner. *On Cryptographic Properties of LFSR-based Pseudorandom Generators*. PhD thesis, Mannheim University, 2004. Available at: `http://th.informatik.uni-mannheim.de/people/zenner/pub/phdzenner.pdf`.