Szyfry strumieniowe w strukturach FPGA

Marcin Rogawski
PROKOM Software S.A.
Rogawskim@prokom.pl

Streszczenie

Konkurs AES, zorganizowany przez NIST, dotyczył wyłącznie szyfrów blokowych – jego celem było zastąpienie przestarzałego standardu, jakim był DES. W latach 2000-2003r. europejskie środowiska uczelniane zorganizowały konkurs NESSIE. Obszar zainteresowań jego uczestników dotyczył wszystkich dziedzin kryptologii. Dzięki temu pojawiły się nowe szyfry, funkcje skrótu, techniki kryptoanalizy, metodologie oceny nowych algorytmów.

W rundzie finałowej konkursu AES znalazło się pięć algorytmów blokowych, które w zasadzie do dzisiaj nie uległy żadnemu praktycznemu atakowi. Rozważano więc, wiele innych kryteriów związanych z praktycznymi ich zastosowaniami. Dopiero głosowanie wyłoniło zwycięzcę – belgijski algorytm Rijndael.

Konkurs AES, oprócz pomnożenia i tak już licznej rodziny szyfrów blokowych, zapoczątkował inne zjawisko – ogromne zainteresowanie tą grupą algorytmów. Do konkursu NESSIE zgłoszono tak dużo nowych szyfrów blokowych, że zostały one podzielone na dwie grupy. Efektem trzech lat pracy środowisk uczelnianych, innych ośrodków naukowo-badawczych oraz osób prywatnych, w ramach NESSIE, były propozycje na nowe standardy: Misty1 (blok 64 bitów) oraz Camellia (blok 128 bitów). Jedyną kategorią algorytmów kryptograficznych, która nie doczekała się żadnych nowych propozycji odpornych na ataki kryptoanalityczne były szyfry strumieniowe. Jednocześnie w dziedzinie tej dokonał się znaczny postęp w kryptoanalizie – dokonano spektakularnych wręcz, praktycznych ataków na najbardziej znany A5, atakom algebraicznym nie oparł się ToyoCrypt, a KSA w RC4 wydaje się również niezbyt bezpieczny (WEP).

Szyfry strumieniowe znalazły swoje zastosowanie we wszelkich bezprzewodowych technologiach. W telefonii GSM dominowały odmiany wspomnianego już szyfru A5, podstawą bezpieczeństwa nowej technologii UMTS jest wprawdzie blokowy Kasumi, ale w trybie strumieniowym. Bluetooth działa z wykorzystaniem E0.

Szyfry strumieniowe niewątpliwie zorientowane są na realizacje programowe – często przy ich implementacjach słyszymy, że generacja jednego bajta strumienia klucza wymaga kilku cykli procesora, podczas gdy dla szyfrów blokowych generacja jednego bajta szyfrogramu zajmuje nawet kilkaset cykli tego samego procesora. Nikt chyba jednak nie wyobraża sobie, że implementacje programowe mogą zastąpić w całości rozwiązania sprzętowe. Telefon komórkowy nie będzie przecież wykorzystywał procesora z 2 Ghz zegarem taktującym. Algorytmy zapewniające poufność w tego typu zastosowaniach korzystają z układów typu ASIC.

W rozwiązaniach sprzętowych szyfry blokowe zawsze będą miały tą przewagę, że można je realizować w wersji potokowej, dzięki której możliwe są realne przepustowości powyżej 1Gb/s. To udogodnienie dotyczy jednak tylko trybu ECB lub prostych trybów licznikowych. Czy więc szyfry strumieniowe mogą rywalizować pod względem przepustowości z szyframi blokowymi działającymi w trybach innych niż ECB ? Jak nowe szyfry strumieniowe: Henkos, Rabbit i polski VMPC prezentują się pod względem efektywności i przepustowości? Celem autora artykułu jest próba oceny możliwości najbardziej znanych oraz najnowszych szyfrów strumieniowych w implementacjach sprzętowych.

1. Wprowadzenie

O algorytmach ...

Najważniejszym usługami zapewniającymi ochronę informacji są: poufność, integralność, niezaprzeczalność i autoryzacja. Funkcje te realizowane są za pomocą różnych algorytmów kryptograficznych. We współczesnej kryptografii możemy wyróżnić dwa rodzaje algorytmów, zapewniających poufność: systemy kryptograficzne symetryczne i systemy z kluczem publicznym (asymetryczne). Jeśli za system kryptograficzny uznamy więc, algorytm szyfrujący i deszyfrujący razem z odpowiednimi kluczami szyfrującym i deszyfrującym, wówczas systemem symetrycznym nazywać będziemy taki, w którym znajomość jednego z kluczy sprowadza się do znajomości obu (w szczególności oba klucze mogą być jednakowe).

W systemach z kluczem publicznym, znalezienie odpowiedniego klucza deszyfrującego (szyfrującego) na podstawie klucza szyfrującego (deszyfrującego) jest problemem trudnym obliczeniowo. Przyjmuje się, że bez dodatkowych informacji (np. rozkładu modułu w RSA) nie jest możliwe w realnym czasie wyznaczenie odpowiadającego klucza szyfrującego (deszyfrującego).

Szyfry symetryczne dzielą się na blokowe i strumieniowe. Algorytmami blokowymi nazywamy takie szyfry, które na podstawie zadanego klucza przekształcają blok wejściowy, na blok o takiej samej długości, w taki sposób, że nie możliwe jest odwrócenie tego przekształcenia. Nie możliwym jest więc odzyskanie bloku wejściowego, na podstawie bloku wyjściowego, bez znajomości klucza przekształcenia. Algorytmami strumieniowymi nazywamy takie, w których operacja szyfrowania (deszyfrowania) polega na przekształceniu za pomocą strumienia znaków. Szyfry strumieniowe dzielą tekst M na bity m_1, m_2, … m_n, a następnie każdy i-ty element jest szyfrowany i-tym elementem strumienia.

Twórcy szyfrów sięgają do idei zawartych w pracy Claude Shannona z 1949r. pt. „Communicaton Theory of Secrecy Systems". W pracy tej po raz pierwszy pojawiły się pojęcia bezpieczeństwa obliczeniowego i bezwarunkowego. Zdefiniowany został jedyny bezwarunkowo bezpieczny szyfr – z kluczem jednorazowym. Autorzy szyfrów blokowych starają się zbliżyć do szyfru – ideału, poprzez naprzemienne stosowanie warstw liniowych (dyfuzja) i nieliniowych (konfuzja). Twórcy szyfrów strumieniowych do zagadnienia konstrukcji algorytmu podchodzą bardziej dosłownie – tworzą algorytmy deterministyczne, których produktem działania ma być bliski losowemu strumień klucza.

Teoria kryptografii często mówi o zatartej granicy pomiędzy tymi rodzinami szyfrów. Szyfr strumieniowy o krótkim okresie można więc uznawać za szyfr blokowy. Weźmy pod uwagę Enigmę. Można ją uważać za szyfr blokowy o długości bloku 26^k lub też jako okresowy szyfr strumieniowy o okresie równym 26^k.(26- ilość liter alfabetu, k - ilość rotorów w maszynie). Szyfr może być więc uważany za szyfr blokowy, gdy każde M_i jest blokiem liter o długości okresu:

$E_k(M) = E_k(M_1)\, E_k(M_2)$ …

lub jako okresowy szyfr strumieniowy, gdy każdy element Mi jest jedną literą, a strumień klucza k jest powtarzany.

Gdy okresy są krótkie szyfr przypomina bardziej słaby szyfr blokowy, gdy wzrasta długość okresu szyfr staje się bardziej podobny do szyfru strumieniowego.

Praca jest poświęcona tej rodzinie algorytmów strumieniowych.

O wymaganiach dla szyfrów symetrycznych ...

Śledząc ostatnie konkursy amerykański AES, europejski NESSIE i japoński CRYPTREC na nowe standardy kryptograficzne nie sposób nie zwrócić uwagi na sposoby oceny algorytmów kryptograficznych. Absolutnie nie wystarcza już tylko udowadniane bezpieczeństwo szyfru czy odporność na znane ataki. Musi on dodatkowo spełniać warunki dotyczące takich zagadnień jak efektywność jego implementacji programowej i sprzętowej, szybkość realizowanej operacji szyfrowania i deszyfrowania, a także adaptacyjność w nietypowych środowiskach.

O układach programowalnych ...

Systemy cyfrowe mogą się opierać o standardowe, uniwersalne elementy wielkiej skali integracji, które mogą być uzupełniane elementami małej i średniej skali integracji. System cyfrowy zbudowany z nich składa się często z wielu odrębnych układów, co powoduje, że drastycznie zwiększają się koszty produkcji takiego rozwiązania. Taka sytuacja spowodowała rozwój nowego podejścia do procesu wytwarzania cyfrowych układów. Preferuje wytworzenie jednego układu scalonego, który spełniałby rolę nawet najbardziej złożonego układu cyfrowego, w jednym, wyspecjalizowanym układzie cyfrowym. Rozwój technologii układów programowalnych spowodował, że układy scalone realizujące nawet bardzo skomplikowane operacje trafiły ze specjalizowanych elektronicznych firm „pod strzechy" domów zwykłych ludzi.

Układy programowalne FPGA (ang. *Field Programmable Gate Array*), umożliwiają zaprojektowanie dużego systemu cyfrowego przez pojedynczego programistę.

Najważniejszymi zaletami systemów mikroprocesorowych wykorzystujących struktury programowalne są:
- łatwość przeprogramowania,
- szybkość realizowanych przez strukturę programowalną operacji,
- odciążenie głównego procesora od operacji najbardziej czasochłonnych,
- mały pobór mocy,
- niska cena.

Układy programowalne są jakby stworzone na potrzeby kryptografii. Każdy może oprzeć swój własny system bezpieczeństwa na zaprogramowanym odpowiednim algorytmem układzie FPGA. Możliwa jest również zmiana z dnia na dzień oprogramowania struktury (np. w przypadku złamania danego algorytmu). Najwięksi twórcy (ALTERA, XILINX) struktur programowalnych czynią kolejne kroki nad bezpieczeństwem konfiguracji (tzw. bitstream) układów FPGA.
Dostępne są już układy zapewniające szyfrowanie plików konfiguracyjnych struktury programowalne.

O programowaniu na układy programowalne …

Autorzy szyfrów strumieniowych bardzo często mówią o szybkości swoich rozwiązań. Ograniczają te informacje tylko do częstotliwości taktowania procesora, do ilości cykli zegarowych, niezbędnych do wytworzenia jednego bajta strumienia oraz do wynikającej z tego przepustowości rozwiązania. Szybkość implementacji programowej zależy nie tylko od tych parametrów, ale także od architektury wybranego procesora oraz czego nie należy lekceważyć – od wyboru kompilatora i jego ustawień. W przypadku porównywania efektywności i szybkości szyfrów strumieniowych należy więc znaleźć dla nich wspólny mianownik. W wyniku konkursu NESSIE powstało opracowanie „Performance of Optimized Implementation of Nessie Primitives", które porównuje szyfry strumieniowe pod każdym względem na różnych platformach w formie implementacji programowej. Porównania rozwiązań sprzętowych szyfrów strumieniowych są dużo rzadsze i na mniejszą skalę.

Wytwarzanie oprogramowania na układy programowalne różni się zasadniczo od programowania na procesory ogólnego przeznaczenia. Przede wszystkim realizując implementacje programowe nie zastanawiamy się nad długością trwania poszczególnych operacji składowych. W rozwiązaniach takich czas trwania „podobnych" operacji – na przykład dodawania 32-bitowych liczb jest równy operacji różnicy symetrycznej na tych samych argumentach. Wszystkie te operacje są wyrównane do najdłużej trwającej operacji w liście rozkazów danego procesora. W procesorach ogólnego przeznaczenia mamy wreszcie stały, a przynajmniej zdefiniowany w jakimś zbiorze wartości, czas trwania pojedynczego cyklu zegarowego.

W rozwiązaniach sprzętowych sposób realizacji implementacji poszczególnych operacji wpływa bezpośrednio na czas trwania tzw. „ścieżki krytycznej". Czas trwania najdłuższej drogi sygnału cyfrowego przez zdefiniowaną przez użytkownika funkcję określa czas trwania cyklu zegarowego. W rozwiązaniach sprzętowych twórca oprogramowania ma większe możliwości realizacji własnych pomysłów. Realizacje algorytmów kryptograficznych, które realizują cały proces szyfrowania jednego bloku danych w trakcie jednego cyklu zegarowego to żadna nowość w przypadku struktur programowalnych (np. wykonanie całego procesu szyfrowania algorytmem AES w ciągu jednego cyklu zegarowego. Wytworzenie 1 bajta szyfrogramu będzie wymagało 1/16 cyklu zegarowego). Więcej na ten temat w rozdziale 2.

Bardzo duże rozbieżności w analizie czasowej tych samych funkcji – w nieco inny sposób zrealizowanych, skłania autora artykułu do wprowadzenia kilku wymagań na implementacje algorytmów strumieniowych. Przede wszystkim wszystkie szyfry strumieniowe, co zrozumiałe, będą implementowane i testowane na jednym rodzaju układów (Altera Flex 10KE). Po za tym większość funkcji, podstawowych składowych, szyfrów strumieniowych zostanie zaimplementowana w postaci biblioteki. Nie pozwoli to na osiągnięcie optymalnych pod względem szybkości realizacji sprzętowych. Proces implementacji wszystkich wybranych algorytmów był dosyć długi i dlatego aby wystrzec się nie obiektywności potrzeba było założenia o jednakowej realizacji tych samych elementów szyfrów strumieniowych (np. LFSR).

Cel pracy …

Sprawdzenie możliwości implementacyjnych, przepustowości i efektywności szyfrów strumieniowych w strukturach programowalnych.

2. Architektury akceleratorów kryptograficznych

O sukcesie w sprzętowej realizacji modułu kryptograficznego decyduje wiele czynników. Możemy je generalnie podzielić na trzy grupy: matematyczne, technologiczne, koncepcyjne. Pierwsze wynikają z doświadczenia projektowego, z wiedzy z dziedziny algebra liniowa oraz z możliwości samego algorytmu kryptograficznego.
Kolejny czynnik, noszący nazwę technologicznego, wynika z faktu najbardziej prozaicznego - ceny układów programowalnych. Oczywiście najłatwiejsze rozwiązanie to zakupić najdroższy układ trzeciej generacji, który oprócz standardowych komórek logicznych posiada wbudowane matryce pamięci oraz zintegrowane bloki DSP. Realizacja kryptograficznego akceleratora jest za pewne najłatwiejsza mając do dyspozycji: doświadczenie, odpowiednią wiedze oraz najnowsze technologie.

Nie mając tego wszystkiego projektant nie jest jednak skazany na porażkę. Zagadnienie wyboru architektury dla modułu kryptograficznego jest często najważniejszym i decydującym zarówno pod względem efektywnościowym jak i przepustowościowym. W poniższym rozdziale są omówione najważniejsze architektury modułów kryptograficznych w sprzętowych akceleratorach.

2.1. Architektura Iteracyjna

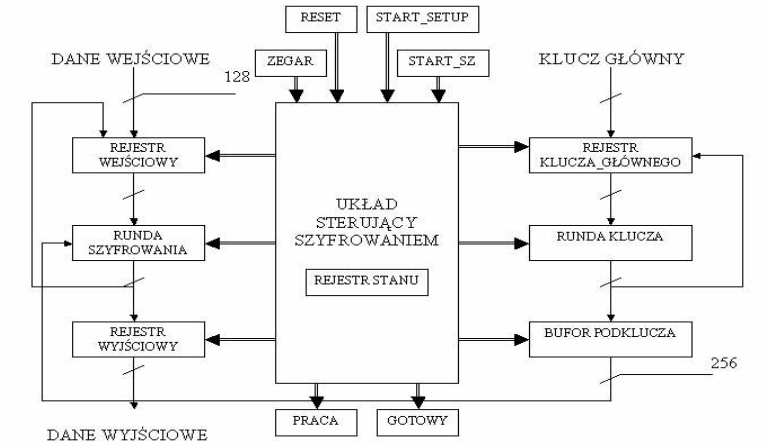
Architektura Iteracyjna jest najnaturalniejszym dla algorytmów kryptograficznych sposobem realizacji. Zarówno funkcje skrótu jak i symetryczna kryptografia preferuje symetrie z punktu widzenia budowy szyfru (rundy w algorytmie są takie same) i rundy (wszystkie bity wchodzącego do rundy bloku są traktowane tak samo). Z tego tez powodu wypływa niezwykle istotny fakt – nie ma potrzeby implementacji całości szyfru, bowiem wystarczy iteracyjne powtórzenie pewnego fragmentu kodu w przypadku rozwiązania programowego lub także iteracyjne przetworzenie sygnału cyfrowego przez dedykowany układ scalony. Z charakteru tego typu implementacji bierze się nazwa architektury.

Głównymi cechami tego typu rozwiązań są przede wszystkim duża oszczędność zasobów środowiska, w którym realizowany jest algorytm. Poprawność realizacji funkcji szyfrowania i deszyfrowania dla rozwiązań programowych nie wymaga dużej ilości komórek pamięci typu RAM, a dla rozwiązań sprzętowych nie dużej ilości komórek pamięci typu ROM. Omawiana cecha architektoniczna nosi nazwę efektywności implementacji i jest zdecydowanie najlepsza w porównaniu z pozostałymi architekturami.

Kalkulacja kosztów, istoty bezpieczeństwa, możliwości przepustowości medium transmisyjnego, do którego przyłączony jest akcelerator kryptograficzny oraz praktyczna wiedza na temat ilości informacji przepływającej przez układ przekłada się na największa popularność tej architektury.

Na całkowity czas procesu szyfrowania bądź deszyfrowania składa się wiec czas propagacji sygnału cyfrowego poprzez logikę zawarta pomiędzy rejestrami, oraz czas zapisu informacji do rejestru. Szybkość przetwarzania tego typu architektury zależy wiec nie tylko od specyfiki algorytmu blokowego, ale także od ilości operacji zapisu informacji przejściowych (np. wartość szyfrogramu po 3 rundach).

Dodać należy w tym miejscu, że architektura ta jako jedyna jest odpowiednia dla szyfrów strumieniowych bez żadnych ograniczeń i wszystkie rozwiązania będą realizowane przy jej użyciu.



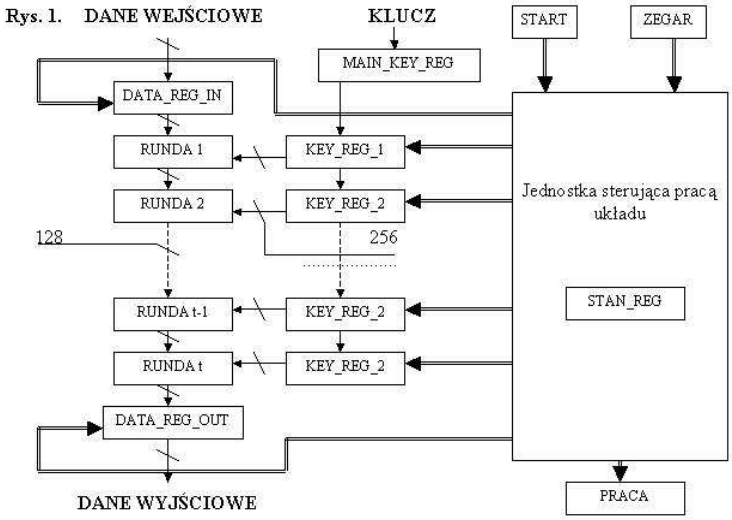
Rys. 2.1. Schemat blokowy architektury Iteracyjnej

2.2. Architektura Kombinacyjna

Innym sposobem realizacji algorytmu kryptograficznego to implementacja postaci funkcji kombinacyjnej danych i klucza, bądź samych danych w przypadku funkcji skrótu. W tego typu rozwiązaniach najistotniejszym faktem jest możliwość realizacji całego procesu szyfrowania/deszyfrowania, obliczania skrótu w ciągu jednego cyklu zegara. Pomijane są zapisy pośrednich wartości, a wszystkie rundy są fizycznie zaimplementowane w układzie scalonym. Prowadzi to oczywiście do bardzo nieefektywnego wykorzystania dostępnych zasobów, ale także wpływa na szybkość przetwarzania dedykowanego rozwiązania. Czas potrzebny sygnałowi cyfrowemu do przejścia „ścieżki logicznej" skraca się. Na czas trwania procesu szyfrowania jednego bloku danych w architekturze iteracyjnej potrzebny jest czas przejścia sygnału cyfrowego przez wszystkie rundy oraz czas zapisu wartości pośrednich do rejestrów wewnętrznych

Dodatkowa zaleta architektury kombinacyjnej jest najmniej skomplikowany układ sterujący. Szybkość przetwarzania układu scalonego zrealizowanego na bazie tej architektury jest wiec doskonałym wskaźnikiem mówiącym o możliwościach implementacyjnych danego algorytmu.

Największa wada oprócz niskiej efektywności jest fakt, ze architektura tego typu jest idealna dla szyfrów w stylu DES. Generacja podkluczy jest w nich bardzo prymitywna, a co za tym idzie czas wyznaczania ich jest pomijalnie mały w stosunku do czasu potrzebnego na realizacje rundy szyfrowania/deszyfrowania. Można powiedzieć, ze dzięki temu wszystkie podklucze rundy są wyznaczone na czas i wynik operacji kryptograficznej jest poprawny.



Rys. 4.2.: Schamat blokowy Architektury Kombinacyjna

Problem ten może być przezwyciężony poprzez zastosowanie wcześniejszej fazy ustawienia podkluczy do wszystkich rund szyfrowania, albo tez wręcz realizacje algorytmu generacji podkluczy poza układem scalonym. Podczas fazy przygotowania modułu sprzętowego wprowadza się podklucze rund, a po zakończeniu tego procesu można rozpocząć właściwe szyfrowanie/deszyfrowanie w ramach architektury kombinacyjnej.

2.3. Architektura Potokowa

Lista rozkazów w procesorach o architekturze potokowej jest bardzo krótka, ale za to bardzo jednolita. Cecha ta pozwala na łatwe i szybkie ich dekodowanie. Inna bardzo istotna cecha jest podział cyklu rozkazowego na 5 etapów: pobranie, dekodowanie, pobranie argumentów, wykonanie, zapisanie wyniku. Każdy z nich jest realizowany przez specjalizowana jednostkę, a czas wykonania każdego z etapów jest bardzo podobny. Architektura potokowa w tego typu procesorach pozwala wiec na wykonywanie kilku rozkazów w ciągu jednego cyklu zegara, dokładniej

zakończenie jednego rozkazu jest realizowane w ciągu jednego cyklu. Problemy, jakie się pojawiają w tego typu rozwiązaniach to zaburzanie potoku przez instrukcje skoków warunkowych.

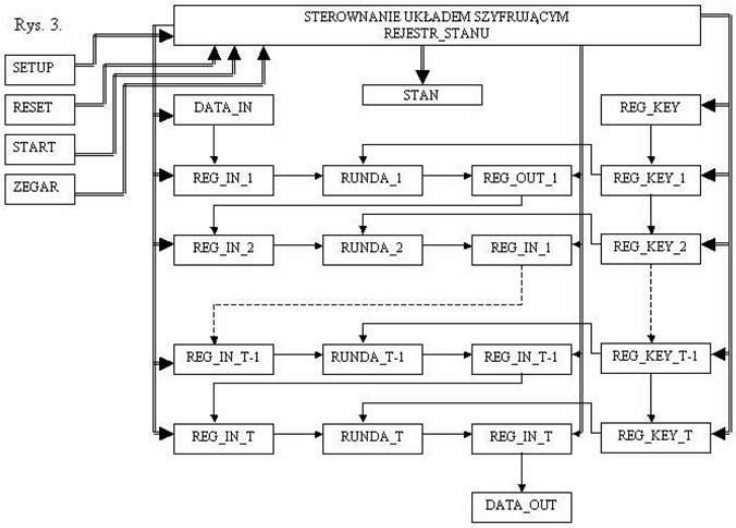
Algorytmy kryptograficzne w większości przypadków składają się z kilku takich samych rund (ewentualnie pierwsza lub ostatnia są nieco inne). Czas propagacji sygnału cyfrowego dla różnych danych dla pojedynczej rundy jest ilość przybliżeniu stały. Można wiec algorytm podzielić na kilka bloków funkcjonalnych, które będą realizowały operacje szyfrowania dana runda. W ten sposób realizowana jest architektura potokowa.

Najważniejszymi zaletami jest ogromna szybkość przetwarzania – rzędu kilku giga-bitów. Tak samo jak w przypadku architektury kombinacyjnej w każdym cyklu realizowane jest szyfrowanie całego bloku danych. Istotna różnica pomiędzy tymi dwoma rozwiązaniami jest fakt, że w przypadku rozwiązania potokowego cykl zegara jest kilka razy krótszy dzięki czemu cały proces szyfrowania/deszyfrowania jest krótszy i oczywiście przetwarzanie jest dużo większe.

Inna bardzo duża zaleta tego typu rozwiązania jest fakt, ze większość szyfrów blokowych można zrealizować w formie potokowej.

Bardzo dużym ograniczeniem jest niezbędna bardzo duża ilość zasobów do realizacji poprawnego procesu szyfrowania i deszyfrowania.

Najpoważniejsza wada jest jednak brak możliwości realizacji innych trybów pracy szyfrów blokowych niż tryb ECB (ang. *Electronic Code Book*). Niemożliwym także jest wykorzystanie tego typu architektury do realizacji szyfrów strumieniowych i funkcji skrótu. Wynika to oczywiści z charakteru tych algorytmów.



Rys. 2.3. Schemat blokowy architektury Potokowej

2.4. Architektury Hybrydowe

W każdej dziedzinie życia często najbardziej przydatnymi rozwiązaniami okazują się być hybrydy. Wiele algorytmów kryptograficznych, takich jak np. Serpent, DES, 3DES, składa się z wielu rund szyfrowania, a jednocześnie implementacja każdej z nich jest na tyle efektywna, że bardziej opłacalnym rozwiązaniem może się okazać realizacja iteracyjno-kombinacyjna.

Najlepszym sposobem realizacji algorytmu Serpent w środowiskach o ograniczonej liczbie zasobów logicznych okazała się architektura iteracyjno – kombinacyjna [3]. W czasie jednego cyklu zegara realizowanych jest aż 8 rund szyfrowania/deszyfrowania. Rozwiązanie to jest nie wiele bardziej zasobo-chłonne, natomiast szybkość przetwarzania jest tam nawet sześciokrotnie większa od rozwiązania iteracyjnego.

Do podobnych wniosków można dojść także w przypadku wielu innych algorytmów kryptograficznych

2.5. Architektury i Algorytmy

	Algorytmy blokowe	Algorytmy strumieniowe	Funkcje skrótu
architektura interacyjna	Każdy algorytm	Każdy algorytm	Każdy algorytm
architektura kombinacyjna	Każdy algorytm, ale czasami niezbędny jest Key Setup	Większość algorytmów, z którymi się zetknął autor można zrealizować używając tej architektury	Większość algorytmów, z którymi się zetknął autor można zrealizować używając tej architektury
architektura potokowa	Praca w trybie ECB, lub prostym licznikowym	____	____
architektura hybrydowa	Dowolne kombinacje	Kombinacyjno-iteracyjne	Kombinacyjno-iteracyjne

Tabela 2.1.: Zestawienie „Architektury i Algorytmy"

3. Implementacja wybranych algorytmów

Szyfry strumieniowe, które zostały przebadane pod względem implementacji:

- **A5**
- **BMGL**(Hastad J., Naslund M.)
- **E0**
- **Helix** (Schneier B.)
- **Leviathan** (Cisco Systems)
- **Lili-128** (Queensland University of Technology, Golic J.)
- **Rabbit** (Cryptico A/S)
- **RC4** (Rivest R.)
- **Sober-16** (Qualcomm)
- **VMPC** (Żółtak B.)
- **W7** (Wave7 Optics, Anagram Laboratories, University of Waterloo)
- **F8** (Matsui M.)
- **Rijnadael – OFB** (J.Deamen, V.Rijmen)

Przepustowość danego rozwiązania – czy to programowego, czy sprzętowego definiujemy jako stosunek ilości przetworzonych (zaszyfrowanych) danych do czasu, jaki potrzebny był na wykonanie tego procesu.

przepustowość = ilość danych / czas na ich przetworzenie

Na wstępie artykułu pisaliśmy o tym, w jaki sposób różni się wytwarzanie oprogramowania na procesory ogólnego przeznaczenia od programowania układów programowalnych. Przypominamy, że bez względu na styl implementacji w procesorach ogólnego przeznaczenia czas trwania jednego cyklu zegarowego jest oczywiście stały. Także stała i ściśle określona jest liczba i rodzaj instrukcji możliwych do wykorzystania na danym procesorze. Programowanie na procesory ogólnego przeznaczenia sprowadza się, zatem do sekwencyjnego wywoływania różnych instrukcji, które razem tworzą program (w szczególności realizujący szyfr strumieniowy)

W implementacjach na struktury programowalne styl pisania wpływa bezpośrednio na maksymalną możliwą częstotliwość taktowania zaimplementowanego rozwiązania. Możliwe jest zrównoleglenie wykonywania różnych operacji. Tak jak już wiemy z rozdziału 2, w strukturach programowalnych jest możliwe zrealizowanie całej, nawet bardzo skomplikowanej operacji w ciągu jednego taktu zegarowego. Wszystko definiuje projektant układu cyfrowego.

Pierwszą cechą szyfrów strumieniowych, która bezpośrednio decyduje o szybkości implementacji programowych i sprzętowych jest długość strumienia klucza powstającego między operacjami uaktualnienia stanu wewnętrznego szyfru. Jak wiadomo, projektanci szyfrów strumieniowych stoją przed dylematem: wzrost długości

produkowanego materiału klucza musi za sobą pociągać, albo zwiększenie ilości elementów stanu wewnętrznego, albo zwiększeniu stopnia skomplikowania przekształceń filtra – którym jest funkcja wyjściowa. Oba podejścia niewątpliwie wpływają na zwiększenie ilości niezbędnych zasobów do wykonania poprawnej implementacji szyfrowania/deszyfrowania. Zawsze jednak szyfry strumieniowe będą miały przewagę nad szyframi blokowymi pod tym względem, że moduł szyfrujący i deszyfrujący mają identyczną budowę – co nie zawsze ma miejsce w przypadku szyfrów blokowych.

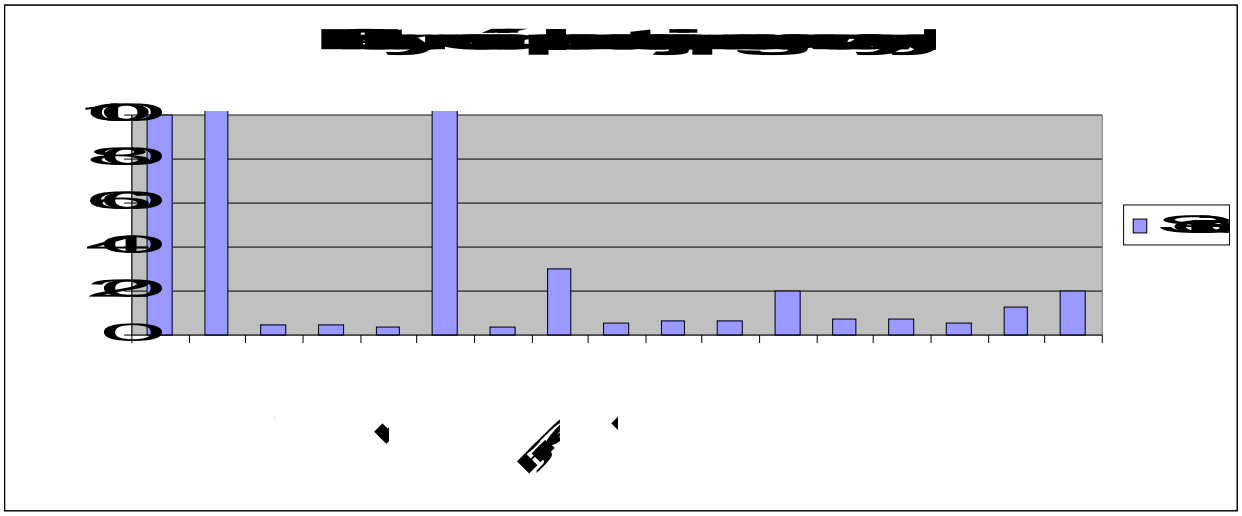
Poniżej znajduje się tabela zestawiająca wybrane algorytmy strumieniowe oraz podstawowe informacje na temat parametrów, o których wcześniej była mowa.

algorytm strumieniowy	długość strumienia klucza	wielkość stanu wewnętrzngo
A5	1	64
BMGL	16	17*128 (2176)
E0	1	128+2 (130)
Helix	32	128
Leviathan	32	128
Lili-128	1	128
Rabbit	128	8*32+8*32+1 (513)
RC4	8	(8*256) 2048
Sober-16	8	128
VMPC	8	(8*256) 2048
W7	8	8*128
F8	64	128
Rijndael – OFB	128	128

Tabela 3.1.: Zestawienie wybranych szyfrów oraz ich wybranych parametrów

W zestawieniu tym brakuje jeszcze dat pojawienia się specyfikacji poszczególnych algorytmów. Zauważylibyśmy wówczas, że twórcy szyfrów strumieniowych coraz bardziej poważnie podchodzą do problemów wydajności. Najwyraźniej widać byłoby to na przykładzie komunikacji bezprzewodowej. Algorytm W7 miał być alternatywą dla A5 w sieci komórkowej GSM. Pomysł jednak nie doczekał się oficjalnego dokumentu RFC i pozostał tylko draftem. Nie mniej jednak prawdopodobnie jest bezpieczniejszym, a już na pewno bardziej efektywnym i szybszym algorytmem niż A5. Trzecia generacja telefonii komórkowej, UMTS, oparła swoje bezpieczeństwo na szyfrze blokowym Kasumi. Oczywiście działa on w zmodyfikowanym trybie strumieniowym OFB i nosi nazwę F8. Architektom systemu UMTS za pewne nie chodziło tylko o fakt posiadanych bardzo dobrych własności sprzętowych algorytmu Kasumi, ale przede wszystkim o to, że praktycznie żaden z ówczesnych algorytmów strumieniowych nie cieszył się swoją odpornością na ataki dłużej niż rok. Widzimy więc rozszerzanie się długości strumienia produkowanego przez szyfry strumieniowe. Wyjątkiem jest tutaj standard Bluetooth, który oparł swoje bezpieczeństwo na E0, które jednorazowo produkuje jeden bit strumienia klucza.

Efektywność implementacji programowych również potwierdza, że najwolniejszymi rozwiązaniami są algorytmy strumieniowe stojące na straży definicji strumieniowości – czyli te produkujące po jednym bicie na jeden stan wewnętrzny generatora pseudolosowego.



Rys.3.1.: Efektywność implementacji programowej

Zoptymalizowane implementacje programowe, najlepszych szyfrów strumieniowych potrzebują poniżej dwudziestu, a nawet dziesięciu cykli zegarowych do wytworzenia jednego bajta strumienia klucza. Szyfry strumieniowe, oparte o binarne LFSR przechodzą do historii i wynika to chyba nie tylko z wyników implementacyjnych. Można takie wrażenie również odnieść czytając komentarze środowisk uczelnianych dotyczących szyfru Lili-128 podczas konkursu NESSIE. Jego struktura została określona jako „staromodna", oparta o binarny LFSR. W zasadzie wszystkie szyfry strumieniowe, które ostatnio zostały przedstawione społeczności kryptologicznej są bardzo efektywne pod względem implementacji programowych. A biorąc pod uwagę, że każdy autor szyfru zakłada, że jego szyfr jest bezpieczny to potrzeba chyba wreszcie zwrócenia uwagi na optymalizację szyfrów strumieniowych pod kątem rozwiązań sprzętowych. Skoro wszystkie są rzekomo bezpieczne i wszystkie bardzo szybkie w implementacjach programowych to przydałoby się dodatkowe kryterium oceny.

Ponieważ autorowi artykułu zależało na obiektywnej ocenie porównywanych szyfrów została stworzona biblioteka funkcji standardowo powtarzających się w wybranych szyfrach. Z jednej strony powoduje to, że osiągnięte wyniki dla poszczególnych szyfrów nie będą optymalne, ale drugiej zaś dzięki temu możliwe będzie zobrazowanie właściwości implementacji sprzętowych poszczególnych szyfrów.

Funkcji tej biblioteki zostały użyte do konstrukcji większości ocenianych szyfrów. Dla porównania z szyframi blokowymi został dodatkowo zaimplementowany szyfr Rijndael w trybie OFB oraz funkcja F8 (Kasumi w OFB).

Wyniki implementacji zawiera poniższa tabela.

	Częstotliwość max.	Zajętość struktury	Przepustowość / zajętość	Przepustowość
	[Mhz]	[LC/mem. bits]	[kb/LC]	[Mb/s]
A5	158,7	299	132,7	39,7
BMGL	166,7	1620 / 40960	26,6	43,1
E0	250	280	178,8	50,0
F8	50	2031	54,3	110,3
Helix	65	2003	126,9	254,2
Leviathan	35	1978	37,9	75,0
Lili-128	204,8	339	114,7	38,9
Rabbit	16,7	4809	7,4	35,6
Rijndael OFB	90	1610 / 40960	254,7	410,0
RC4	70,4	459 / 4096	122,7	56,3
Sober-16	25,18	525	85,3	44,8
VMPC	71,9	514 / 12228	86,3	44,4
W7	161,3	674	478,6	322,6

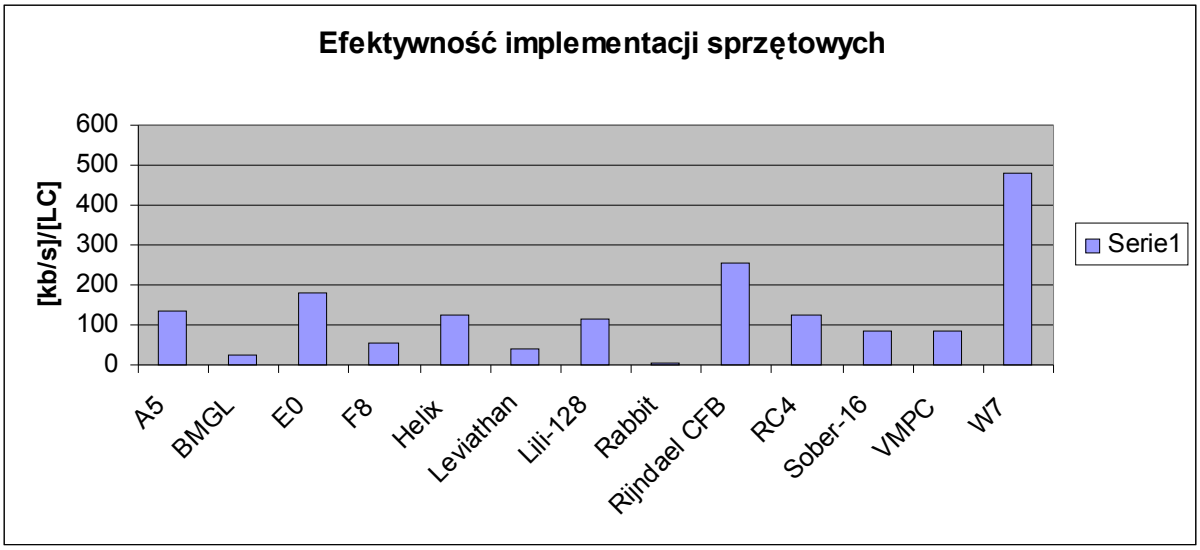
Tabela 3.2.: Zestawienie wyników implementacji szyfrów strumieniowych

Rys.3.2.:Porównanie przepustowości poszczególnych algorytmów strumieniowych

Ciekawą, a jednocześnie niezbyt pozytywną informacją dla szyfrów strumieniowych jest fakt, że algorytmem o najlepszej przepustowości okazał się blokowy Rijndael w trybie OFB. Dalsze w kolejności były: W7, który operuje na LFSR, którego elementami są bajty, a nie nieefektywne LFSR oparte o bajty oraz Helix, który swoją filozofią konstrukcji przypomina nieco Serpenta (ang. *many simple rounds*).

Jedno z twierdzeń kryptologii mówi, że mając dobry szyfr blokowy mamy dobry szyfr strumieniowy oraz dobrą funkcje skrótu. Niestety dla szyfrów strumieniowych to twierdzenie działa tylko w jednym kierunku. Twórcy telefonii komórkowej UMTS nadzwyczaj mocno wzięli sobie to do serca i większość funkcji kryptograficznych oparli o blokowego Kasumi.

Bardzo często porównywana jest efektywność implementacji rozwiązań sprzętowych. Definiuje się ją jako stosunek przepustowości do ilości potrzebnych zasobów – w implementacjach na struktury programowalne komórkach logicznych. Poniższy wykres przedstawia wyniki osiągnięte dla wybranych algorytmów.



Rys.3.3.:Efektywność implementacji na układy programowalne

4. Podsumowanie i wnioski

Czy implementacje algorytmów kryptograficznych są ważne dla przeciętnego użytkownika systemów utajniających ? Nie, dopóki samo szyfrowanie i deszyfrowanie jest dla nich transparentne i nie zauważa, że:

- strony www otwierają się wolniej,
- cały system komputerowy działa wolniej,
- bardzo długo łączy się z bankiem,
- dochodzi do niego przerywana rozmowa.

Poza tym porównywanie możliwości implementacyjnych ma największe znaczenie dla inżynierów wykonujących systemy utajniające. A ponieważ kryptologia już dawno przestała być nauką czysto teoretyczną i jeżeli jakieś zagadnienie nie jest praktyczne i stosunkowo trudno się je realizuje, a dodatkowo nie jest ono efektywne to takie rozwiązanie jest odrzucane.

W teorii szyfrowania symetrycznego zawsze doszukiwano się podobieństw między szyframi strumieniowymi a blokowymi. Co ciekawe do dzisiaj istnieje pogląd, że granica pomiędzy nimi jest bardzo płynna i dany algorytm można rozpatrywać w różnych kontekstach i może on być rozpatrywany zarówno jako blokowy, jak i strumieniowy (np. Enigma).

Co ciekawe implementacje sprzętowe w dosyć wyraźny sposób kreślą granice pomiędzy rodzinami algorytmów. Szyfry strumieniowe, bardzo wydajne w rozwiązaniach programowych są dużo wolniejsze w rozwiązaniach sprzętowych i bardziej swoim charakterem przypominają funkcje skrótu niż szyfry blokowe.

Wydaje się, że bardzo niekorzystną dla szyfrów strumieniowych sytuacją jest fakt, że nie istnieje, żaden standard międzynarodowy definiujący szyfrowanie strumieniowe.

W rozwiązaniach sprzętowych najszybszym okazał się algorytm blokowy w trybie strumieniowym, a najbardziej znane algorytmy strumieniowe są dużo wolniejsze od W7, który nie ma żadnego znaczenia w zastosowaniach praktycznych.
Pozostałe algorytmy, za wyjątkiem szyfru Helix uzyskały podobne przepustowości – około 50Mb/s, co oczywiście na dzisiejsze potrzeby rozwiązań bezprzewodowych jest wystarczające.

5. Literatura

[1]. www.altera.com
[2]. Bora Piotr, Czajka – „Implementation of the Serpent Algorithm using Altera devices" -2000r http://www.csrc.nist.gov/CryptoToolkit/aes/round2/comments/20000513-pbora.pdf
[3].Bora Piotr – „Metody sprzętowej implementacji algorytmów kryptograficznych" – 2003r.
[4]. Boesgaard M., Pedersen T., Vesterager M., Zenner E. – "The Rabbit stream cipher –Design and Security Analysis" – dostępny na stronie http://www.cryptico.com/Files/Filer/SASC_Rabbit.pdf

[5]. Czerwiński Remigiusz – "Analiza Implementacji algorytmu Misty-1 w strukturach programowalnych"
[6]. Elbrit AJ, Yip W., Chatwynd B., Paar C. – „An FPGA Implementation and Performance Evaluation" of the AES Block Cipher Candidate Algorithm Finalists
[7]. Fips197 - http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
[8]. Fluhrer S., Lucks S. – "Analysis of E0 Encrypton System"

[9]. Gaj Kris., Lien R., Grembowski T., - „A 1 Gbit/s Partially Unrolled Architecture of Hash Function Sha-1 and Sha512" – 2004r.

[10]. Hastad J., Naslund M. – "BMGL: Synchronous Key-stream Generator with Provable Security" -2000r. dostępny na stronie http://www.cryptonessie.org
[11]. Hawkes P., Rose G. – "Primitive Specyfication and Supporting documents for Sober-t16 Submission to NESSIE" – 2002r. – dostępny na stronie http://www.cryptonessie.org
[12]. Helix - http://www.schneier.com/paper-helix.pdf

[13]. Kijowski Mariusz – "Ataki korelacyjne na szyfry strumieniowe" –Praca dyplomowa WAT 2003r.
[14]. Kiviharju M. – "Algebraic Attacks and Stream Ciphers" – 2004r.

[15]. Łuba Tadeusz – „Synteza Układów Cyfrowych”- 2003r.
[16]. Łuba Tadeusz – Wykłady dotyczące dekompozycji funkcjonalnej (www.zpt.pw.waw.pl)
[17]. Mańk Krzysztof. – cykl wykładów z przedmiotu „Szyfry Strumieniowe”
[18]. McGrew D., Fluher S. (Cisco Systems) – “The stream cipher Leviathan. Specification and Supporting documents” – 2000r. dostępny na stronie http://www.cryptonessie.org
[19]. Menezes A.– „Handbook of Aplied Cryptography”
[20]. Misztal Michał. - cykl wykładów z przedmiotu “Projektowanie szyfrów blokowych”
[21]. NESSIE „Performance report” dostępny na stronie http://www.cryptonessie.org
[22]. NESSIE - „Security report” dostępny na stronie http://www.cryptonessie.org
[23]. Rc4 - http://www.wisdom.weizmann.ac.il/~itsik/RC4/rc4.html
[24]. Rogawski Marcin. – “Architektury akceleratorów kryptograficznych opartych na układach logicznych FPGA” -2004r. dostępny na stronie www.kryptografia.com
[25]. Rogawski Sebastian. – “Kryptograficzna ochrona informacji w systemie komórkowym UMTS ” – praca magisterska na Politechnice Gdańskiej 2005r.
[26]. Schneier Bruce – „Kryptografia dla praktyków”- 1994r.
[27]. Simpson L., Dawson E., Golić J., Millan W. – “Lili128 – Keystream Generator” – dostępny na stronie http://www.cryptonessie.org
[28]. Szmidt Janusz., Misztal Michał. – „Wstęp do Kryptologii”-2002r.
[29]. Thomas S., Anthon D., Berson T., Gong G. - “The W7 Stream Cipher Algorithm” – 2002r. – internet deaft draft-thomas-w7cipher-02.txt
[30]. Wobst Reinhardt. – „Kryptologia. Budowanie i Łamanie zabezpieczeń”

[31]. Żółtak Bartosz - “VMPC One-way function and Stream Cipher” - 2004r.