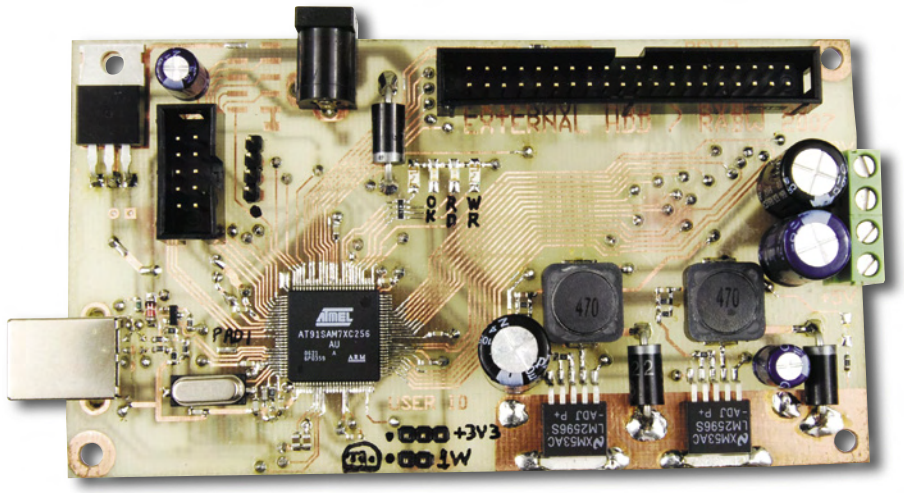
Przejściówka USB–ATA z szyfrowaniem danych AES – bezpieczny dysk twardy, część 1

AVT–5139

Ostatnio wiele się mówi a to o zaginionych laptopach polityków, a to o zniszczonych dyskach w ich komputerach. Jak pamiętamy, głośno było również o zatopieniu notebooka w wannie. Czy były to tylko przypadki, czy nieudolne próby zakamuflowania danych zapisanych na dyskach twardych?

Rekomendacje:

projekt będzie szczególnie przydatny dla tych, którzy mają coś do ukrycia na swoich komputerach.



PODSTAWOWE PARAMETRY

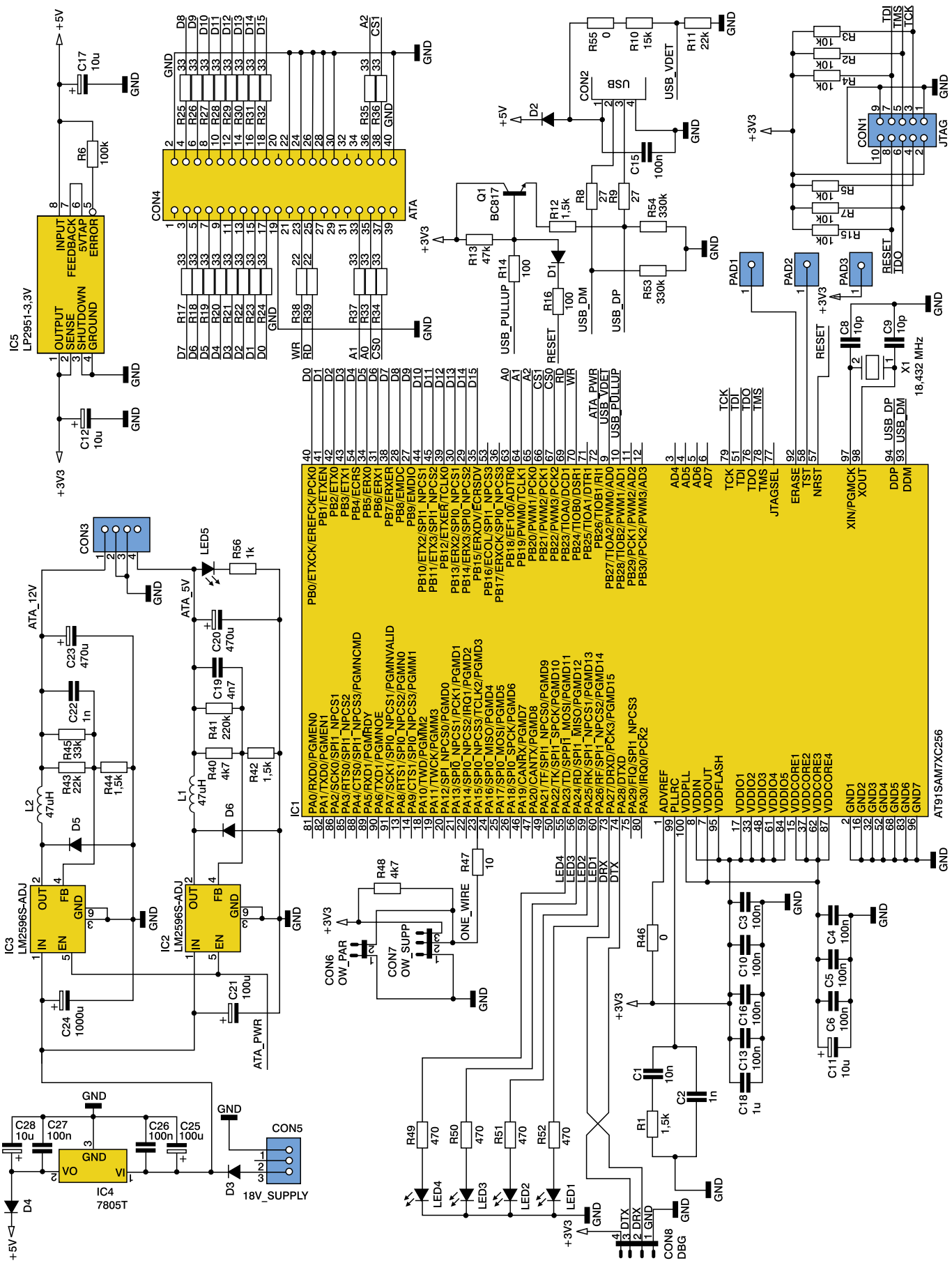
- Płytka o wymiarach 122,5x68 mm
- Zasilanie: 18 V z zewnętrznego zasilacza niestabilizowanego
- Obsługiwane dyski twarde: Parallel ATA, do 2 TB
- Interfejs: USB 2.0, Full Speed
- Szyfrowanie danych: AES, 128 bit
- Klasa urządzenia: Mass Storage Class (pamięć masowa USB), nie są wymagane dodatkowe sterowniki pod Windows 2k i XP
- Rzeczywista szybkość transmisji danych: około 0,4...0,5 MB/s
- Mikrokontroler: AT91SAM7XC256 (ARM7TDMI)
- Typowe zastosowania: zabezpieczenie prywatnych danych przed niepowołanym dostępem

Obecnie wejść w posiadanie zewnętrznego dysku twardego z interfejsem USB nie jest trudno: mamy do dyspozycji tanie przejściówki USB–IDE (USB–ATA), możemy także sami skonstruować podobne urządzenie na układach firm Cypress lub Texas Instruments (opisywane już na łamach EP). Przedstawiona tutaj przejściówka jest jednak inna niż te opisane do tej pory, czy dostępne w handlu: ma bowiem możliwość szyfrowania i deszyfrowania danych zapisywanych na dysku. Należy wspomnieć, że szyfrowanie to jest obecnie jednym z najmocniejszych dostępnych dla „zwykłych śmiertelników" standardów w tej dziedzinie, a mowa tutaj o AES, czyli *Advanced Encryption Standard*. Podanie specjalnego klucza służącego do „otwarcia" dysku następuje w wyjątkowo prosty sposób: poprzez przyłożenie dowolnego urządzenia z interfejsem 1–Wire zawierającego unikatowy numer seryjny.

Niestety mimo, iż złamanie zabezpieczeń przedstawionego tutaj urządzenia jest bardzo trudne (omówione pod koniec artykułu), muszę lojalnie ostrzec Czytelników o podstawowym mankamencie przejściówki, jakim jest niska (a jak na układy dostępne w handlu nawet bardzo niska) szybkość transmisji danych, wynosząca około 0,5 MB/s, czyli ≈4 Mbit/s.

Wszystkie dane, które komputer PC ma zapisać na dysk, są najpierw szyfrowane z użyciem dostarczonego z zewnątrz klucza, a potem (już zaszyfrowane) zapisywane są na dysk twardy podłączony do przejściówki. Wszystkie dane, które komputer chce odczytać z dysku, są natomiast odczytywane, deszyfrowane i wtedy dopiero odsyłane do komputera. Tym sposobem urządzenie dla komputera jest w pełni przezroczyste.

Odszyfrowanie zawartości dysku twardego, na którym znajdują się dane zakodowane w standardzie AES jest zadaniem praktycznie niewykonalnym, a co najmniej nieopłacalnym, jeśli nie zna się odpowiedniego kodu klucza. Oczywiście, gdy spróbujemy taki dysk odczytać



Rys. 1. Schemat elektryczny

bezpośrednio za pomocą zwykłej przejściówki USB–ATA lub typowo umieścimy dysk w komputerze, system nie wykryje na nim jakichkolwiek ważnych (w znaczeniu „mających sens") danych. Także wszelkie próby odtworzenia klucza (nawet znając algorytm, jakim dane zostały zakodowane) są nieopłacalne pod względem czasu i kosztów.

Advanced Encryption Standard

Teoria AES bazuje na bardzo potężnym podłożu matematycznym (teoria Galois [1]), którego opisanie wykracza poza ramy tego artykułu. Założeniem moim było opisanie projektu w sposób jak najbardziej praktyczny, z tego względu o AES powiemy tylko to, co jest niezbędne do zrozumienia zasady działania urządzenia.

Algorytm AES cechuje się tym, iż jest jawny i dostępny dla wszystkich zainteresowanych – oficjalną specyfikację AES każdy może pobrać ze strony [2]. Mimo iż sam algorytm szyfrowania i deszyfrowania jest powszechnie znany, AES pozostaje skuteczną metodą zabezpieczania informacji. Dlaczego? Jest tak dlatego, że „siła" AES tkwi w trudności odgadnięcia klucza, czyli zwykłej liczby, od której zależy sam algorytm. W przypadku tutaj opisanego dysku, jako klucz zastosowano ciąg danych (liczbę) o długości 128 bitów. Standard AES dopuszcza także jeszcze bezpieczniejsze klucze o długościach 192 i 256 bitów.

AES należy do grupy szyfrów z tzw. kluczem symetrycznym. Oznacza to, że ten sam klucz potrzebny będzie do zaszyfrowania i odszyfrowania danych. Jest to bardzo wygodna właściwość, ponieważ czyni urządzenie bardziej „przejrzystym" pod względem działania.

Kolejną właściwością szyfrowania AES jest relatywnie prosta implementacja algorytmu: albo software'owa (w tym C, C++, Java), albo sprzętowa (VHDL) – dostępnych jest wiele gotowych bibliotek na rozmaitych licencjach. Linki do przykładowych implementacji znajdują się w artykule z Wikipedii dotyczącym AES [3].

Zasada działania

Schemat elektryczny przejściówki szyfrującej przedstawiono na **rys. 1**. Szyfrowanie AES jest zrealizowane w pełni sprzętowo, za pomocą układu peryferyjnego mikrokontrolera AT91SAM7XC256 (IC1). Dla optymalnego wykorzystania szybkości działania modułu kryptograficznego, do szyfrowania bloków danych wykorzystano zintegrowany z tym modułem kanał DMA (PDC – *Peripheral DMA Controller*). Szczegóły działania modułów AES i TDES znajdują się w nocie katalogowej mikrokontrolera [4].

Za komunikację urządzenia z dyskiem twardym odpowiada programowo zrealizowany interfejs ATA (równoległy), napisany w oparciu o standard ATA/ATAPI–6 [6]. Fizyczne połączenie dysku i mikrokontrolera zapewnia moduł PIO (*Parallel Input–Output Controller*). Najmłodsze 16 bitów portu PIOB stanowi szynę danych interfejsu ATA (D15...D0), natomiast niektóre z pozostałych sygnałów kontrolera PIOB wykorzystane są do generowania sygnałów sterujących ATA: A2...A0, CS1, CS0 (wybór rejestru dysku), WR i RD (sygnały strobujące odpowiednio zapisu i odczytu).

Rezystory znajdujące się przy złączu interfejsu ATA (R17...R39) mają wartości zalecane w specyfikacji ATA/ATAPI [6], lecz jest to jedynie formalność. Wartości tych rezystorów mają stanowić dopasowanie impedancji przy dużych przepływnościach (a co za tym idzie częstotliwościach), jakie występują na taśmie interfejsu ATA podczas transferów DMA. W niniejszym projekcie transfery DMA dla ATA nie są wykorzystywane, więc rezystory te można zastąpić zworami.

Z racji, że oprogramowanie zostało napisane wg specyfikacji ATA/ATAPI–6, obsługiwane są zarówno dyski twarde zgodne ze standardami ATA/ATAPI wcześniejszymi niż wersja 6 (zwane dalej „starymi" dyskami), jak i „nowe", o pojemnościach rzędu kilkuset gigabajtów, zgodne ze standardami ATA/ATAPI–6 i wyższymi. Maksymalny rozmiar dysku obsługiwanego przez przedstawione urządzenie to 2 TB, mimo iż specyfikacja ATA/ATAPI–6 definiuje maksymalny rozmiar napędu do 144 PB [6] (144 petabajty = 134217728 GB). Ograniczenie do „jedynie" 2 TB dotyczy urządzeń pracujących jako *USB Mass Storage Device Class*, ponieważ zastosowany w nich protokół przezroczystości dla poleceń SCSI [5] narzuca ograniczenia zawarte z kolei w samym SCSI: adresowanie odbywa się 4–bajtowo, czyli można zaadresować 2^{32} bloków (tutaj sektorów), a każdy po 512 B, co razem daje dokładnie 2048 GB = 2 TB. Zainteresowanych szczegółami zachęcam do lektury [5].

Dyski „nowe" akceptują polecenia odczytu i zapisu: READ SECTORS EXT (kod polecenia 24h) i WRITE SECTORS EXT (34h) oraz READ SECTORS (20h) i WRITE SECTORS (30h). Dyski „stare" akceptują jedynie polecenia 20h i 30h. Różnica pomiędzy tymi poleceniami jest m.in. taka, że 24h i 34h posługują się adresowaniem 48–bitowym (omówionym wyżej), a 20h i 30h adresowaniem 28–bitowym (obsługa dysków do 128 GB). Przy starcie i inicjalizacji interfejsu ATA urządzenie rozpoznaje więc, jaki dysk został podłączony po tym czy napęd zgłosi błąd (COMMAND ABORTED) przy próbie odczytu jednego sektora poleceniem 24h. Metoda może nie jest zbyt elegancka, lecz zawsze skuteczna. Jeśli przejściówka wykryje „nowy" dysk, wszystkie operacje zapisu i odczytu odbywają się przy pomocy poleceń 48–bitowch (EXT).

Obsługa interfejsu USB i klasy magazynującej odbywa przy pomocy zmodyfikowanego kodu omówionego w artykule „Czytnik kart SD" [5] zoptymalizowanego do dużych ilości danych. Funkcje wywoływane najczęściej zadeklarowano jako *inline*, co wpłynęło na zwiększenie szybkości transmisji danych pomiędzy komputerem PC i dyskiem twardym. Zastosowano także 8–kilobajtowy bufor dla danych pomiędzy USB i ATAPI, przez co dane te mogą być odczytywane i zapisywane po 16 sektorów za jednym wywołaniem polecenia READ SECTORS (EXT)/ WRITE SECTORS (EXT) [6] (jeden sektor dysku ma rozmiar 512 B).

Fizyczne podłączenie mikrokontrolera do portu USB komputera PC realizowane jest poprzez standardowy układ pośredniczący wzorowany na podanym w nocie katalogowej mikrokontrolera [4]. Układ ten składa się z elementów R8...

R14, R16, R53...R55, Q1, D1. Sterowanie włączeniem podciągnięcia linii DP do +3,3 V odbywa się przez rezystor R12 zwierany tranzystorem Q1 do szyny zasilania. Włączenie rezystora podciągającego powoduje rozpoczęcie transmisji danych od hosta do urządzenia (w prezentowanym projekcie transmisja rozpoczyna się dopiero po podaniu klucza). Dzielnik rezystancyjny R10, R11 zapewnia sygnał informujący mikrokontroler o tym, czy host jest aktualnie podłączony do urządzenia i czy jest on zasilany.

Jak zostało wspomniane na początku artykułu, „klucz do dysku" podawany jest przed startem urządzenia za pomocą dowolnych układów scalonych z interfejsem 1–Wire. Do tego celu przewidziano dwa (właściwie zdublowane) złącza: CON6 i CON7. Urządzeniami 1–Wire generującymi klucz mogą być m.in. popularne „pastylki Dallasa" (czyli układy *iButton* z interfejsem 1–Wire, np. DS1990). Oczywiście mogą to być nawet termometry, np. DS18B20, a chyba najtańszym rozwiązaniem będzie DS2401 (tylko numer seryjny w plastikowej obudowie). Warunkiem jest tylko to, aby układ 1–Wire miał swój unikatowy numer seryjny.

O aktualnym statusie urządzenia informują diody LED1...LED4. Dioda LED1 oznacza gotowość do odczytu układu 1–Wire. Dioda LED2 oznacza poprawnie wprowadzony numer seryjny (odczytany bajt CRC układu 1–Wire zgodny z obliczonym). Diody LED3 i LED4 sygnalizują operacje odpowiednio odczytu i zapisu dysku twardego. Dioda LED5 sygnalizuje obecność napięcia +5 V zasilającego dysk twardy.

Pozostałe elementy hardware'u są typowe dla większości urządzeń z układami SAM7, czyli: filtr PLL (R1, C1, C2) i rezonator kwarcowy X1 wraz z kondensatorami C8, C9. Złącze JTAG (CON1) posiada wszystkie sygnały jak standardowe złącze JTAG dla SAM7, lecz zrezygnowano z niektórych powtarzających się wyprowadzeń. Dodatkowo wyprowadzone jest złącze modułu DBGU (CON8), w projekcie nie wykorzystywane.

Twardy dysk dołączony do przejściówki zasilany jest ze stabilizatorów impulsowych IC2 i IC3. Jeśli Czytelnik zamierza podłączyć napięcie zasilania znacznie odbiegające od 18 V (np. o około 6 V), lub zamierza zastosować inne elementy bierne dla stabilizatorów impulsowych IC2 i IC3, powinien zajrzeć do noty katalogowej układów LM2596, aby ustalić ich wartości. Wersja przedstawiona na schemacie dostosowana jest do napięcia wejściowego około 18 V. Napięcie zbliżone 18 V najłatwiej uzyskać kupując lub budując prosty zasilacz niestabilizowany (elementy: transformator o znamionowym napięciu na uzwojeniu wtórnym 12 V i prądzie obciążenia około 2 A, mostek prostowniczy i kondensator wygładzający przebieg). Dobrym pomysłem może też być zastosowanie zasilacza do laptopa.

Wartości rezystorów ustalających napięcie dla stabilizatorów LM2596S–ADJ (R40...R45) dobrane są tak, aby na wyjściu uzyskać napięcia jak najbardziej zbliżone do +5 V i +12 V używając najłatwiej dostępnych i jak najtańszych rezystorów. Niewykluczone, że będzie konieczna korekta wartości tych rezystorów.

Dioda D3 (głównie będąca zabezpieczeniem przed odwrotną polaryzacją napięcia wejściowego) w prototypie jest diodą Shottky'ego, lecz jedynie dlatego, że wtedy w układzie występuje jeden typ diod na większy prąd. Bez obaw zamiast diody Shottky'ego w miejsce D3 można zastosować zwykłą diodę o odpowiednim prądzie przewodzenia (I_F) lub nawet zastąpić ją zworą, rezygnując tym samym z zabezpieczenia przed odwrotną polaryzacją napięcia zasilania.

Stabilizator IC4 (zwykły 7805) pełni funkcję elementu pośredniego pomiędzy wysokim (18 V) napięciem zasilania całego urządzenia a niskim napięciem 3,3 V zasilającym mikrokontroler – na nim wydziela się ciepło, które nie mogłoby się wydzielić na samym stabilizatorze IC5 (LP2951). Dzięki tej „strategii" uzyskano

WYKAZ ELEMENTÓW

Rezystory

R46: 0 Ω 0805
R55: 0 Ω 1206
R47: 10 Ω 0603
R38, R39: 22 Ω 0603
R8, R9: 27 Ω 0603
R17...R32, R34...R37: 33 Ω 0603
R14, R16: 100 Ω 0603
R49...R52: 470 Ω 0603
R56: 1 kΩ 0603
R1, R12, R42, R44: 1,5 kΩ 0603
R40, R48: 4,7 kΩ 0603
R2, R3, R4, R5, R7, R15: 10 kΩ 0603
R10: 15 kΩ 0603
R11, R43: 22 kΩ 0603
R45: 33 kΩ 0603
R13: 47 kΩ 0603
R6: 100 kΩ 0603
R53, R54: 330 kΩ 0603
R41: 220 kΩ 0603

Kondensatory

C8, C9: 10 pF 0603
C2, C22: 1 nF 0603
C19: 4,7 nF 0603
C1: 10 nF 0603
C3, C4, C5, C6, C10, C13, C15, C16, C26, C27: 100 nF 0603
C18: 1 µF/16 V 0805
C11, C12, C17: 10 µF/16 V tantalowy, rozmiar A
C28: 10 µF/16 V elektrolityczny przewlekany
C21, C25: 100 µF/25 V elektrolityczny przewlekany
C20, C23: 470 µF/16 V elektrolityczny przewlekany (najlepiej low ESR)
C24: 1000 µF/25 V elektrolityczny przewlekany

Półprzewodniki

D1, D2, D4: 1N4148 SMD
D3: 1N5822 lub podobna wg opisu
D5, D6: 1N5822
LED1...LED5: diody LED SMD 0805 lub 1206
Q1: BC817
IC1: AT91SAM7XC256
IC2, IC3: LM2596S–ADJ
IC4: LM7805 TO–220
IC5: LP2951 3,3 V, SO–8

Inne

X1: rezonator kwarcowy 18,432 MHz SMD
L1, L2: 47 µH 20%, 1,9 A, 0,15 Ω
CON1: złącze 2x5pin (opcja, JTAG)
CON2: gniazdo USB–B przewlekane
CON3: Terminal Block, 4pin
CON4: goldpin 2x20pin, najlepiej „box header"
CON5: złącze zasilania DC
CON6: goldpin 2pin
CON7: goldpin 3pin
CON8: goldpin 4pin

węzeł (nóżka 8, wejście stabilizatora IC5), na którym utrzymuje się napięcie bliskie 5 V (dokładnie 5–0,7 V przez spadki napięć na diodach 1N4148). Napięcie to może być przydatne przy prowadzeniu własnych eksperymentów z przejściówką.

Interfejs 1–Wire i wersje oprogramowania

Jeden układ 1–Wire ma numer seryjny o długości 64 bitów (oczywiście razem z polami CRC i family code, więc „właściwy" numer seryjny stanowi jedynie 48–bitów). Wszystkie 64 bity numeru seryjnego wykorzystane są do generowania „połowy" klucza (przypominam, że klucz ma długość 128 bitów). Jeśli dostarczymy jedną połowę klucza, a drugą potraktujemy jako stałą, możemy ułatwić pracę osobie podejmującej próby złamania zabezpieczeń naszego dysku – osoba ta może także czytać Elektronikę Praktyczną i znać przedstawiony tutaj projekt... Z tego powodu przygotowałem dwie wersje oprogramowania oznaczone jako *AES_DISK_V1* i *AES_DISK_V2* różniące się poziomem bezpieczeństwa.

Wersja *V1* jest łatwiejsza w użyciu lecz mniej bezpieczna: pierwsza część klucza jest stała i wynosi 0125F1ED9FC1BABEh, a druga połowa kodu klucza otrzymywana jest z numeru seryjnego układu scalonego Dallas. Takie postępowanie znacznie zmniejsza liczbę kombinacji kluczy i ułatwia „włamanie się" do dysku bezpośrednio przez interfejs ATA.

Wersja *V2* jest mniej przyjazna użytkownikowi, ponieważ musi on w odpowiedniej kolejności przyłożyć dwa układy firmy Dallas do styków złącza CON6 lub CON7 przejściówki. I oczywiście z układu pierwszego zostanie wygenerowana pierwsza część klucza, a z układu drugiego – druga. Jest to bardziej „męczące" lecz zapewnia dużo większe bezpieczeństwo danych niż wersja pierwsza.

Robert Brzoza–Woch
rabw@poczta.fm

Literatura

[1] http://www.partow.net/projects/galois/
[2] http://www.csrc.nist.gov/publications/fips/fips197/fips–197.pdf
[3] http://en.wikipedia.org/wiki/Advanced_Encryption_Standard
[4] http://www.atmel.com/dyn/resources/prod_documents/doc6209.pdf lub prościej: www.atmel.com i w okienku wyszukiwania „AT91SAM7XC256"
[5] R. Brzoza–Woch „Czytnik kart SD", EP 4/2007
[6] AT Attachment with Packet Interface – 6 (ATA/ATAPI–6) Rev. 3B, www.t13.org
[7] http://www.truecrypt.org/
[8] http://www.cryptosystem.net/aes/