

# Konwersja hasła na klucz - standard PKCS5

<http://ipsec.pl/kryptografia/konwersja-hasla-na-klucz-standard-pkcs5.html>

{Dokumenty [ja href="http://www.rsasecurity.com/rsalabs/pkcs/"](http://www.rsasecurity.com/rsalabs/pkcs/) PKCS<sub>i</sub>/<sub>a<sub>j</sub></sub> tworzone na przestrzeni lat przez firme RSA DSI stały sie podstawa wielu standardów kryptograficznych, akceptowanych później przez organizacje miedzynarodowe.

{Ten konkretnie dokument, PKCS 5, określa sposób szyfrowania danych za pomoca hasła. Problem ten, na pozór banalny (bierzemy szyfr, tekst jawny i szyfrujemy), jest tak na prawde powaznym problemem implementacyjnym, który skompromitował wiele aplikacji. Przykładem niech bedzie tzw. czeski atak na format OpenPGP.

{Podstawowymi pytaniami na jakie należy odpowiedzieć sa:

- {jaki szyfr zastosować (strumieniowy czy blokowy)?
- {jeśli blokowy, to w jakim trybie (CBC, CFB, ...)?
- {czy i jak zapewnić ochrone integralności (CRC-32 czy może coś silniejszego)?
- {jaki format pliku wynikowego przyjąć?

{Standard PKCS 5 odpowiada na wiekszość tych pytań i warto go stosować jako wzorcowa implementacje w każdym przypadku, kiedy musimy zapisać porcje danych zaszyfrowanych hasłem.

{PKCS 5 do tego celu wykorzystuje szyfr blokowy w trybie CBC z ochrona integralności. To ostatnie jest bardzo istotne - jak pokazuje historia, brak silnej ochrony integralności przy najlepszym nawet szyfrowaniu zaowocował skompromitowaniem protokołu SSHv1 oraz wspomnianego wyżej OpenPGP.

{PKCS 5 zaleca wykorzystanie modyfikatora klucza (salt) aby zapewnić, że ten sam tekst jawny zaszyfrowany wiele razy tym samym kluczem da zawsze inny kryptogram.

{Istotnym elementem PKCS 5 jest sposób przygotowania klucza dla szyfru blokowego. Oryginalna specyfikacja PKCS 5 w wersji 1.5 wykorzystuje pojedynczy DES z kluczem 56 bitów, co jest rozwiązaniem słabym i archaicznym. W poniższym opisie oprzemy sie o 3DES z kluczem 168 bitów, co jest rozwiązaniem standardowym i godnym zaufania, zgodnym z wersja 2.0 standardu.

{W PKCS 5 [niej](#) wykorzystujemy wprost hasła użytkownika do szyfrowania danych. Jest ku temu kilka powodów:

- {hasło może być za krótkie lub za długie dla 3DES, który potrzebuje dokładnie 24 bajtów
- {hasło może nie być zbyt wysokiej jakości (łatwe do zgadnięcia), w związku z czym musimy dołożyć starań by to zgadywanie utrudnić

{Procedura generowania klucza dla 3DES jest w skrócie następująca:

1. {Do hasła użytkownika dołączany jest {salt
2. {Z wyniku obliczany jest skrót SHA1 i ta operacja może być powtarzana dowolna ilość razy (patrz poniżej)
3. {Z wynikowego skrótu wybieramy 24 bajty klucza dla 3DES oraz 8 bajtów wektora początkowego ([ja href="http://echelon.pl/leksykon/iv.php"](http://echelon.pl/leksykon/iv.php) <sub>i</sub>IV<sub>i</sub>/<sub>a<sub>j</sub></sub>) dla trybu CBC

{Operacja z punktu 2. może być powtarzana dowolna ilość razy - to znaczy z hasła i {saltu obliczamy skrót, z niego kolejny skrót i tak dalej. Jest to konieczne w przypadku, gdy pierwszy skrót (160 bitów dla SHA1) nie wystarcza na pełny klucz 3DES (168 bitów). Wówczas bierzemy 160 bitów z pierwszego skrótu, i pierwsze 8 z drugiego. Następne 8 przeznaczamy na wektor

początkowy (IV). Po drugie, wielokrotne powtarzanie iteracji (nawet do kilkuset razy) może być praktycznie niezauważalne podczas szyfrowania i deszyfrowania pliku przez legalnego użytkownika, może natomiast o te kilkaset razy zwiększyć czas potrzebny intruzowi do zgadnięcia hasła metoda słownikowa.

{Pamiętajmy, że zarówno salt jak i wektor początkowy muszą być znane odbiorcy, aby mógł on poprawnie rozszyfrować wiadomość. Są one jawne, więc można je dołączyć po prostu do wiadomości w oznaczonych miejscach. Zaletą powyższej procedury jest jednak to że nie trzeba dołączać przynajmniej wektora, bo odbiorca posiada wszystkie informacje niezbędne do jego odtworzenia (hasło i {salt}).

{Problem, do którego prędzej czy później dochodzi podczas szyfrowania CBC to dopełnianie ({padding}) - co zrobić w sytuacji, gdy nasze dane nie wypełniają do końca ostatniego bloku kryptogramu? W przypadku DES zachodzi to zawsze, jeśli długość wiadomości nie jest wielokrotnością 8. Według standardu bloki takie należy wypełnić w sposób, który do dziś określa się nazwą {PKCS 5 padding, a który wywodzi się z <http://echelon.pl/leksykon/rfc1423.txt> i RFC 1423 [i/a](#).

{Załóżmy, że nasze dane mają długość 38 bajtów, należy je zatem zapisać w 5 blokach DES po 8 bajtów. Ostatni blok będzie zawierał 6 bajtów danych i 2 bajty niewykorzystane, które trzeba wypełnić. PKCS 5 zaleca w tym miejscu wypełnienie ich dwoma dwójkami, czyli bajtami o wartości 0x02. Analogicznie, jeśli zostanie nam 7 bajtów, wypełniamy je siedmioma bajtami o wartości 0x07. Jeśli nasze dane mają długość będącą wielokrotnością 8, to musimy dodać do nich jeden pełny blok zawierający osiem bajtów 0x08. W ten sposób wypełnienie jest zawsze jednoznaczne i wiadomo gdzie kończą się dane, a ile bajtów stanowi wypełnienie - wystarczy popatrzeć na ostatni bajt ostatniego bloku.

{Sprawa nie do końca uregulowana jest ochrona integralności zaszyfrowanych danych. W opisie trybu <http://echelon.pl/leksykon/cbc.php> i [CBC](#) [i/a](#), wspomniałem niektóre ataki, które są możliwe do przeprowadzenia jeśli dane zaszyfrowane tym (lub innymi) trybem są pozbawione ochrony integralności. PKCS 5 zaleca wykorzystanie <http://echelon.pl/leksykon/mac.php> i [MAC](#) [i/a](#) i wspomina o <http://echelon.pl/leksykon/hmac.php> i [HMAC](#) [i/a](#), jednak nie mówi wprost jak i gdzie należy je stosować.

{Wystrzegać się należy słabych kryptograficznie sum kontrolnych w rodzaju <http://echelon.pl/leksykon/crc.php> i [CRC](#) [i/a](#). Najlepszym rozwiązaniem jest zastosowanie funkcji <http://echelon.pl/leksykon/hmac.php> i [HMAC](#) [i/a](#) obliczanej dla wynikowego kryptogramu (nie dla wejściowego tekstu jawnego), która jest następnie dołączona do niego w postaci jawnej.

{Pytanie o kolejność operacji - czy najpierw szyfrować, potem liczyć skrót czy odwrotnie - pojawiało się wielokrotnie przy okazji implementacji rozmaitych protokołów kryptograficznych. Wymieniona niżej praca Hugo Krawczyka pokazuje, że obliczanie skrótu dla tekstu jawnego może prowadzić do osłabienia całej ochrony i poprawnym jest liczenie skrótu dla wynikowego kryptogramu. [h2](#) [i Bibliografia](#) [i/h2](#)

- {PKCS 5: Password Based Cryptography <http://citeseer.ist.psu.edu/5205.html> i <http://citeseer.ist.psu.edu/>
- {Hugo Krawczyk, {„The Order of Authentication and Encryption”, <http://citeseer.ist.psu.edu/krawczyk/>