KRYPTOGRAFIA

WPROWADZENIE

KRYPTOGRAFIA

Czym jest kryptografia? — "Zabezpieczenie dostępu do informacji"

Steganografia — ukrywanie informacji w innej informacji

Jak rozumieć zabezpieczanie? Generalnie są to przynajmniej 3 poziomy:

- **poufność** (**confidentiality, privacy**)
- **integralność** — odczytane dane są takie same jak były zapisane, **uwierzytelnianie** (**integrity, authenticity**) — udowodnienie możliwości dostępu do danych
- **dostępność** (**availability**) — dane są dostępne kiedy są potrzebne

Jak rozumieć ataki?

- **pasywny** - podsłuchiwanie (**eavesdropping**, Eve)
  (Alice [m] –Eve sees [m]–> Bob gets [m])
- **aktywny** - ingerencja w komunikację (**malicious**, Mallet, Mallory)
  (Alice [m] –Mallet changes[m] into [n]–> Bob gets [n])

BEZPIECZEŃSTWO

Nowoczesna kryptografia vs klasyczna kryptografia

- **kryptografia symetryczna** — kryptografia z kluczem tajnym (secret-key, private-key, shared-key) — szyfrowanie i odszyfrowanie wykorzystuje ten sam klucz
- **kryptografia asymetryczna** — kryptografia z kluczem publicznym — szyfrowanie i odszyfrowanie będzie korzystać z różnych kluczy i nie da się wyliczyć jednego z drugiego; Pozwala dodatkowo na uwierzytelnianie.

Inne pojęcia związane z kryptografią

- **uwierzytelnianie** — uwierzytlnia tylko użytkownika
- **podpis elektroniczny** — uwierzytelnia osobę i dokument
- **funkcje haszujące** — wyznacza skrót / haszuje nieodzyskalnie dane
- **pseudolosowość** — deterministyczny algortym wyliczający udający losowość

Szyfrowanie symetryczne — przestrzeń tekstów (wiadomości do zaszyfrowania) $\mathcal{M}$ wraz z trzema algorytmami:

- **generowanie klucza** (**Gen**)
- **szyfrowanie** (**Enc**)
- **deszyfrowanie** (**Dec**) — deterministyczna część — tekst musi zostać odzyskany do swojego oryginalnego stanu

Nazywamy to pojęcie ogólne **kryptosystemem** w tym przypadku **symetrycznym**

Trzy algorytmy muszą spełniać poniższe własności

- **Gen** — algorytm probabilistyczny, wyznacza klucz k zgodnie z pewnym rozkładem prawdopodobieństwa
- **Enc** — na wejściu ma tekst $m \in \mathcal{M}$ oraz klucz k a na wyjściu zwraca szyfrogram c ze zbioru $\mathcal{C}$, symbolicznie $\mathrm{Enc}_k(m)$
- **Dec** — na wejściu ma szyfrogram c oraz klucz k a na wyjściu zwraca wiadomość jawną m, symbolicznie $\mathrm{Dec}_k(c)$
- Szyfrowanie musi spełniać poniższą własność dla każdego m i k:

  $$\mathrm{Dec}_k(\mathrm{Enc}_k(m)) = m$$

  (zaszyfrowana a potem odszyfrowana wiadomość jest równa oryginalnej wiadomości)

Zbiór wszystkich kluczy k nazywamy przestrzenią kluczy $\mathcal{K}$.
Używając szyfrowania symetrycznego, klucz k jest wybierany losowo z przestrzeni kluczy $\mathcal{K}$
Protokół wymiany wiadomości z wykorzystaniem powyższych pojęć może wyglądać następująco:

1. Alicja i Bob wybierają wspólny klucz k stosując algorytm Gen
2. Jeżeli nadawca Alicja chce przesłać wiadomość m do Boba, szyfruje ją kluczem k, oblicza szyfrogram $c = \mathrm{Enc}_k(m)$ i przesyła c otwartym kanałem.
3. Odbiorca Bob deszyfruje szyfrogram c kluczem k, otrzymujac $m = \mathrm{Dec}_k(c)$

Oznacza to problem z przesłaniem klucza do obiorcy — jak Bob ma otrzymać klucz?

Bezpieczeństwo komunikacji

Atakująca Ewa znając Dec i k może odzyskać m na podstawie podsłuchanego komunikatu c, zatem obie strony muszą trzymać klucz k w sekrecie. Nie muszą one ukrywać Dec, mówi o tym **prawo Kerchoffa**, szyfrowanie musi być tak zaprojektowane, że nawet gdy atakujący pozna sposób deszyfrowania i tak nie będzie mógł odszyfrować podsłuchanego szyfrogramu. **Całe bezpieczeństwo spoczywa na kluczu** k.

Klasyczne szyfry pracowały na literach a nie bitach (np. szyfr Cezara opierający się na **permutacji** — zmianie kolejności wartości w ciągu; gdyby szyfr Cezara miał permutację dla każdej litery, byłby nie do złamania)

Jak zdefiniować że dany kryptosystem jest bezpieczny?

- Możemy różnie definiować bezpieczeństwo w zależności od potrzeb i założeń praktycznych
- Możemy ustawić wymagania bezpieczeństwa kryptosystemu na różnym poziomie. Atakujący nie może wyznaczyć:
  1. klucza k
  2. tekstu jawnego m z szyfrogramu c
  3. fragmentu tekstu jawnego m
  4. żadnej informacji na temat tekstu jawnego m
- Na bazie tego możemy wprowadzić kilka modeli bezpieczeństwa. Standardowe stosowane dziś w analizie bezpieczeństwa to:
  - atak ze znanym szyfrogramem (**ciphertext-only attack**)
  - atak ze znanym tekstem jawnym (**known-plaintext attack**)
  - atak z wybranym tekstem jawnym (**chosen-plaintext attack**) (atakujący wybiera co potrzebuje, best case np wszystkie litery)
  - atak z wybranym szyfrogramem (**chosen-ciphertext attack**)

  Wszystkie są równo ważne, zależą od sposobu wykorzystania szyfru.

W nowoczesnej kryptografii wymagamy również ścisłych matematycznych założeń. **Łatwiej założyć, że dany problem matematyczny jest trudny obliczeniowo niż założyć, że złożony kryptosystem jest bezpieczny.** Definicje i założenia pozwalają na przeprowadzanie ścisłych dowowdów bezpieczeństwa systemów.
Wiele nowoczesnych kryptosystemów opiera się na **dowodliwym bezpieczeństwie** (**provable security**), zależy od przyjętych definicji i założeń. Nie jest ono równoważne **rzeczywistemu bezpieczeństwu** (**real-world security**), ponieważ mogą się różnić przyjętymi definicjami.
Mówimy też często o **bezpieczeństwie obliczeniowym** (**computional security**), które zależy od aktualnego stanu wiedzy i technologii (np. wprowadzanie nowych standardów)

ONE-TIME PAD

Istnieją też systemy **bezwarunkowo bezpieczne** (**unconditional security / perfectly secret**), których bezpieczeństwo nie zależy od posiadanych mocy obliczeniowych.
Przykładem jest **One-Time Pad**, nazywany **szyfrem Vernama**.

Definicja — Kryptosystem $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ o przestrzeni tekstów $\mathcal{M}$ nazywamy bezwarunkowo bezpiecznym, gdy dla każdego rozkładu prawdopodobieństwa tekstów z $\mathcal{M}$, każdej wiadomości $m \in \mathcal{M}$ oraz każdego szyfrogramu $c \in \mathcal{C}$ takiego, że $\Pr[C = c] > 0$ zachodzi

$$\Pr[M = m \mid C = c] = \Pr[M = m].$$

Powyższa definicja opisuje sytuację ataku ze znanym szyfrogramem, podsłuchanym szyfrogramem. Jeżeli znamy szyfrogram, to nie możemy powiedzieć jaki był oryginalny tekst nie znając klucza. Wiedza o szyfrogramie nie zdradza żadnych informacji o tekście jawnym.
Wzór z definicji mówi, że prawdopodobieństwo (a posteriori) wysłania pewnej wiadomości $m \in \mathcal{M}$, pod warunkiem, że podsłuchano konkretny szyfrogram, jest takie samo jak prawdopodobieństwo (a priori) wysłania wiadomości m

- Z jakiegokolwiek tekstu może powstać każdy szyfrogram i viceversa.
- Brak możliwości weryfikacji jaki tekst odszyfrowany jest prawidłowy (z perspektywy atakującego)

Upraszczając powyższą definicję możemy przyjąć równoważnie:
Kryptosystem $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ o przestrzeni tekstów $\mathcal{M}$ nazywamy bezwarunkowo bezpiecznym wtedy i tylko wtedy, gdy dla każdych dwóch wiadomości $m, m' \in \mathcal{M}$ oraz każdego szyfrogramu $c \in \mathcal{C}$ takiego, że $\Pr[C = c] > 0$ zachodzi

$$\Pr[\mathsf{Enc}_K(m) = c] = \Pr[\mathsf{Enc}_K(m') = c].$$

- Prawdopodobieństwo że z jakiegokolwiek tekstu wyjdzie ten sam szyfrogram jest takie samo

Szyfr OTP 1917r patent Vernam — nie było od razu wiadomo że jest on bezwarunkowo bezpieczny. 25 lat później potwierdzony przez Shannon'a

Sposób działania OTP:

- Dla ustalonej wartości $l > 0$, przestrzeni tekstów $\mathcal{M}$, przestrzeni kluczy $\mathcal{K}$ oraz przestrzeni szyfrogramów $\mathcal{C}$ zdefiniowanych jako $\{0,1\}^l$, czyli ciągi binarne długości l:
- **Gen** – algorytm generowania kluczy wybiera $k \in \mathcal{K} = \{0,1\}^l$ z jednostajnym rozkładem prawdopodobieństwa. Zbiór zawiera 2^l kluczy, zatem prawdopodobieństwo każdego klucza k wynosi 2^{-l}.
- **Enc** – dla każdego klucza $k \in \{0,1\}^l$ oraz $m \in \{0,1\}^l$, algorytm zwraca szyfrogram

  $$c = k \oplus m,$$

  gdzie operacja $\oplus$ oznacza różnicę symetryczną (ang. **bitwise exclusive or**, **XOR**). (XOR daje 1/2 szans na 1 i 1/2 szans na 0)
- **Dec** – dla każdego klucza $k \in \{0,1\}^l$ oraz $c \in \{0,1\}^l$, algorytm zwraca wiadomość

  $$m = k \oplus c.$$

W uproszczeniu — każdy bit ma swój własny klucz, szyfrogram jest długości wiadomości oraz klucz nie jest używany ponownie. **Brak praktyczności** wynika z tego jak dużo danych trzeba przesłać.

OTP jest bezwarunkowo bezpieczny pod warunkiem że klucz musi:

- być losowy, nie pseudolosowy
  - problem skąd czerpać losowe bity
- mieć długość taką samą jak tekst jawny
- musi być jednorazowy, nie można go użyć powtórnie
- musi mieć takiej samej długości klucz i wiadomość

Na podstawie OTP można wyprowadzić wniosek, że każdy bezwarunkowo bezpieczny szyfr musi mieć klucz długości przynajmniej takiej jak tekst jawny. Co oczywiście oznacza, że używając kryptosystemów w których klucze są krótsze niż wiadomości nigdy nie uzyskamy bezwarunkowego bezpieczeństwa.

Szyfrowanie symetryczne

Bezpieczeństwo obliczeniowe

W praktyce możemy przyjąć, że niewielka ilość informacji może "wyciekać", przy ograniczonych mocach obliczeniowych. Możemy ustalić poziom prawdopodobieństwa takiego zdarzenia przy ustalonej mocy obliczeniowej, np. 2^{-128}, 2^{-1000} (przeglądając wszystkie bity osiągniemy ostatecznie poprawny klucz odszyfrowujący całą wiadomość — bardzo bardzo mała szansa na odnalezienie drugiego klucza, który da sensowną wiadomość)

W praktyce można złamać szyfr przy nieograniczonych mocach obliczeniowych Ogólnie przyjęto pojęcie **bepzieczeństwa obliczeniowego** (**computation security**), które zależy od aktualnego stanu wiedzy i technologii. Jest to jednak mało precyzyjne.

Możemy rozważyć 2 podejścia do definicji:

- Konkretne — ustalamy ograniczenie na prawdopodobieństwo sukcesu atakującego w określonym czasie (czyli przy pokreślonych mocach obliczeniowych) (używana raczej w praktycznej kryptografii)
- Asymptotyczne — dla określonego parametru bezpieczeństwa n, przykładowo dla długości klucza (używana raczej w teoretycznym podejściu do kryptografii)

Bezpieczeństwo konkretne

W pierwszym przypadku definiujemy bezpieczeństwo kryptosystemu na podstawie dwóch parametrów t oraz ε.

Definicja (bezpieczeństwo konkretne). Kryptosystem jest (t, ε)-bezpieczny, jeżeli atakującemu uda się po czasie co najwyżej t złamać system z prawdopodobieństwem co najwyżej ε.

Można przyjąć ogólnie, że dla klucza n-bitowego, czyli przestrzeni kluczy $|\mathcal{K}| = 2^n$, atak brutalny powiedzie się z prawdopodobieństwem $c \cdot t \cdot 2^{-n}$ dla pewnej stałej c.

Przykład. Dla $c = 1$ oraz $n = 60$, procesor $4\,\mathrm{GHz}$, wykonując $4 \cdot 10^9$ cykli na sekundę, potrzebuje $\frac{2^{60}}{4 \cdot 10^9}$ sekund, czyli około 9 lat.
Superkomputer wykonujący $2 \cdot 10^{16}$ operacji na sekundę będzie potrzebował około jednej minuty. Dla $n = 80$ ten sam superkomputer będzie już potrzebował około 2 lat.

Dziś w praktyce wymagamy minimum $n = 128$ ale zaczyna być realnie potrzebny 256.
Wiek wszechświata szacuje się na 2^{58} sekund.

Bezpieczeństwo asymptotyczne

W analizie szyfrów trudno jest stosować podejście konkretne, ponieważ zmienia się moc obliczeniowa komputerów. Stosujemy zwykle drugie podejście, asymptotyczne do szacowania poziomu bezpieczeństwa Wykorzystujemy w nim dwa pojęcia:

- **algorytm o czasie wielomianowym** (są szybsze niż wykładnicze)
- **funkcję zaniedbywalną** (**negligible function**)

Funkcja $f\colon \mathbb{N} \to \mathbb{R}$ jest zaniedbywalna, jeżeli dla każdego dodatniego wielomianu p istnieje liczba całkowita $N > 0$ taka, że dla wszystkich $n > N$ zachodzi

$$f(n) < \frac{1}{p(n)}.$$

Można to wykorzystać w przypadku dużych wartości n, prawdpodobieństwo sukcesu atakującego będzie wówczas pomijalnie małe.

(W uproszczeniu — jeżeli funkcja zbliża się do 0 wykładniczo i jest lepsza niż $1/n^k$ to jest zaniedbywalna)

Algorytm probabilistyczny A ma **wielomianową złożoność czasową** (**Probabilistic Polynomial-Time**, PPT), jeżeli istnieje wielomian p taki, że dla każdego n-bitowego wejścia x, algorytm $A(x)$ zakończy się po co najwyżej $p(n)$
Jeżeli algorytm A nie jest wielomianowy to dla odpowiednio dużego n, szybko stanie się niepraktyczny w użyciu dla atakującego

Kryptosystem jest bezpieczny asymptotycznie jeżeli dla każdego algorytmu PPT stosowanego przez atakującego A i dla każdego dodatniego wielomianu p istnieje taka liczba naturalna N że dla każdego $n > N$ prawdopodobieństwo sukcesu A jest mniejsze niż $1/p(n)$
Definicja uproszczona - Kryptosystem jest bezpieczny asymptotycznie jeżeli dla każdego algorytmu PPT prawdopodobieństwo sukcesu atakującego A jest zaniedbywalne

Nierozróżnialność

Ważne pojęcie w analizie bezpieczeństwa szyfrów (**indistinguishability**)

Nierozróżnialność definiujemy jako próbę odgadnięcia przez atakującego który z dwóch tekstów został zaszyfrowany znając jedynie jego szyfrogram. Możemy opisać to jako eksperyment, w którym atakujący A przygotowuje dwie wiadomości $m_0, m_1 \in \mathcal{M}$. Jedna z nich, wybrana losowo z jednostajnym rozkładem prawdopodobieństwa, jest szyfrowana losowym kluczem. Otrzymany szyfrogram jest następnie przesyłany do A. Atakujący musi teraz odgadnąć, która wiadomość została zaszyfrowana.

Kryptosystem uznajemy za **idealnie nierozróżnialny**, jeżeli żaden atakujący nie jest w stanie odpowiedzieć poprawnie z prawdpodobieństwem większym od 1/2

Definicja nierozróżnialności mówi, że kryptosystem jest bezpieczny, gdy nie ma algorytmu PPT który atakujący może wykorzystać do odgadnięcia która wiadomość została zaszyfrowana z prawdopodobieństwem znacząco wyuższym niż zgadywanie losowe z prawdopodobieńswem 1/2.

Kryptosystem symetryczny jest nierozróżnialny przez podsłuchującego (EAV-bezpieczny) jeżeli dla każdego algorytmu PPT stosowangego przez atakującego A istnieje zanidebywalna funkcja (...). Taka definicja obejmuje tylko jeden szyfrogram jednego z dwóch tekstów jawnych, jest bardzo ograniczona. W rzeczywistości podsłuchujący zazwyczaj może podsłuchać więcej szyfrogramów. Definicja systemu EAV-bezpiecznego została rozszerzona do bardziej praktycznej wersji z **wieloma zaszyfrowanymi wiadomościami** (**Multiple Message Security - MMS**) W tym przypadku zamiast pary wiadomości m0 i m1, stosowane są dwie listy wiadomości o tej samej długości M0 i M1, po wyborze b szyfrowana jest lista Mb do listy szyfrogramów C. Atakujący wybiera b prim odpowiadający liście zaszyfrowanych wiadomości.

Atak z wybranym tekstem jawnym

Bardzo ważną własnością bezpiecznych szyfrów jest zabezpieczenie przed **atakiem z wybranym tekstem jawnym** (**Chosen Plaintext Attack CPA**). W przypadku MMS atakujący wszystkie wiadomości musiał wybrać na samym początku, w przypadku CPA może wybierać je w trakcie eksperymentu. W poniższym eksperymencie wykorzystujemy pojęcie "czarnej skrzynki", wyroczni (**oracle**, która szyfruje wiadomości na wejściu nieznanym atakującemu kluczem, używając probabilistycznego algorytmu $\mathsf{Enc}_k(\cdot)$. Atakujący może korzystać z wyroczni wielokrotnie.
Uproszczenie — Atakujący może szyfrować swój wybrany tekst

Standard szyfrowania symetrycznego

W zastosowaniach praktycznych możemy wykkorzystywać różne szyfry symetryczne, uznawane w danym momencie za bezpieczne. W szczególności polecany standard szyfrowania symetrycznego — aktualnie **AES** (**Advanced Encryption Standard**). Szyfr AES jest szyfrem blokowym, czyli permutacją pseudolosową zbioru tekstów, czyli ciągów bitowych postaci $\{0,1\}^n$, gdzie n jest długością bloku tekstu.

Permutacja f zbioru $\{0,1\}^n$ przekształca tekst jawny w szyfrogram o tej samej długości n. Wszystkich możliwych tekstów mamy 2^n, zatem f może być jedną z $(2^n)!$ możliwych permutacji zbioru ciągów długości n.
Szyfr blokowy jest permutacją z kluczem, czyli funkcją

$$F\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n.$$

Przykładowo działanie szyfru opisuje

$$F_{\mathsf{key}}(m) = F(\mathsf{key}, m) = c,$$

gdzie klucz ma długość k bitów, a blok tekstu ma długość n bitów. W przypadku szyfru AES mamy $n = 128$ oraz $k \in \{128, 192, 256\}$.
Funkcje F_{key} oraz F_{key}^{-1} są szybkie obliczeniowo dla danego klucza key.

Twórcy szyfru AES Vincent Rijmen i Joan Daemen już w propozycji szyfru napisali i wykazali, że szyfr AES jest nieodróżnialny od permutacji losowej.

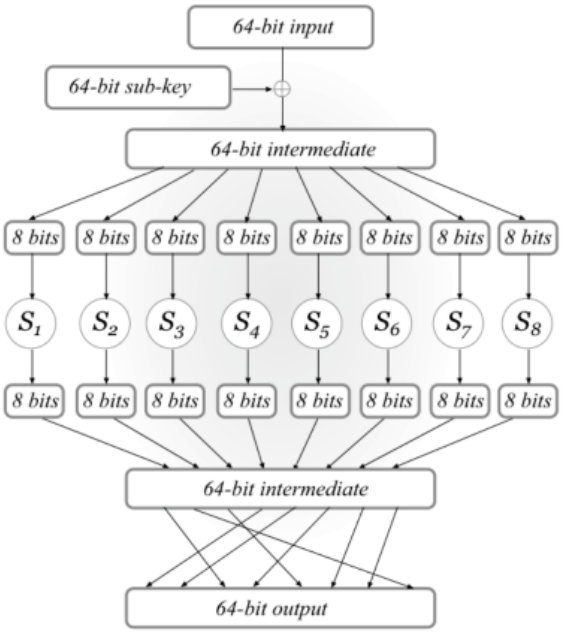
W przypadku szyfrów blokowych, również szyfru AES, rozważane są trzy rodzaje ataków na nieodróżnialność:

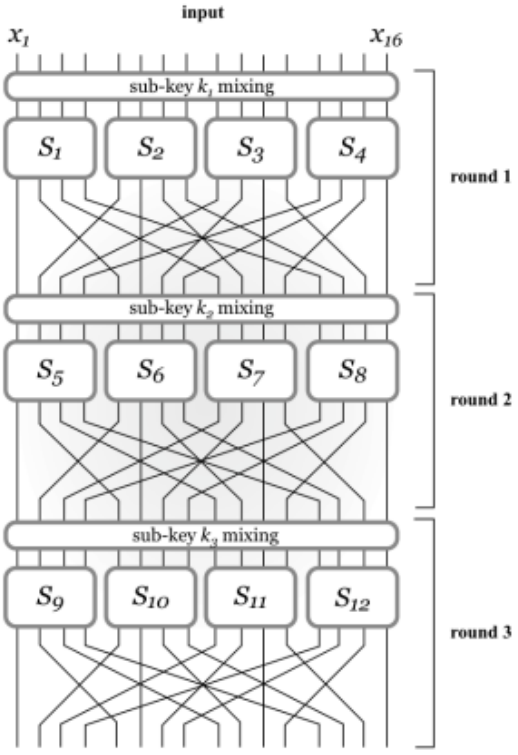
- **atak ze znanym tekstem jawnym** (**known-plaintext attack, KPA**) — atakujący zna pary wejść/wyjść, dla pewnego nieznanego klucza oraz zbioru tekstów z nieznanego źródła
- **atak z wybranym tekstem jawnym** (**chosen-plaintext attack, CPA**) — atakujący posiada szyfrogramy, dla pewnego nieznanego klucza oraz zbioru wybranych przez atakującego tekstów
- **atak z wybranym szyfrogramem** (**chosen-ciphertext attack, CCA**) — atakujący posiada szyfrogramy, dla pewnego nieznanego klucza oraz zbioru wybranych przez atakującego tekstów oraz dla wybranych szyfrogramów

Dodatkowo oprócz ataków na odróżnienie rozważne są analogiczne klasy ataków, których celem jest wyznaczenie nieznanego klucza key użytego w F_{key}. **W praktyce szyfr blokowy musi zachowywać się jak permutacja losowa**. Przyjmując długość bloku równą n bitów, mamy $2^n!$ permutacji, konkretną permutację n bitową można zapisać na $\log\big((2^n)!\big) \approx n \cdot 2^n$ bitów. Dla $n > 20$ jest to już niepraktyczne, a dla $n > 50$ niewykonalne. Pamiętajmy, że AES oraz inne współczesne szyfry używają $n = 128$. Oznacza to, że musimy znaleźć uproszczony, zalgorytmizowany sposób zapisania działania szyfru (działającego jak losowa permutacja tekstów). W takiej konstrukcji wymagamy by dwa działania F_{key} na dwóch blokach, które różnią się jednym bitem dawały dwa (prawie) całkowicie niezależne wyniki. Prawie, bo niemogą być równe, nie można wtedy odszyfrować szyfrogramu. Z tego wynika, że zmiana jednego bitu bloku wejścia do F_{key}, dla nieznanego losowego klucza key, również powinno dawać (prawie) niezależne wyniki. A to finalnie oznacza, że zmiana jednego bitu tekstu powinna wpływać na wszystkie bity szyfrogramu (**efekt lawinowy**). Zauważmy, wpływać ale nie zmieniać, może się zmienić z prawdopodobieństwem 1/2.

Jednym ze sposobów opisania w prosty sposób działania przypominającego działanie losowej permutacji, jest wykorzystanie sieci **substytucji-permutacji** (**Substitution-Permutation Network, SPN**). Szyfry działają w rundach w których wykonywane są trzay rodzaje działań:

- mieszanie z kluczem: $x = x \oplus k$, gdzie k jest podkluczem w rundzie, wyznaczonym na podstawie klucza key;
- podstawianie: $x = S_1(x_1) \parallel S_2(x_2) \parallel \cdots \parallel S_p(x_p)$, gdzie x_i są kolejnymi bajtami x;
- przestawianie, czyli permutacja bitów x, w wyniku działania rundy.





AES jest blokowym szyfrem rundowym działającym w 10, 12 lub 14 rundach w zależności od długości klucza. W każdej rundzie wykonywane są 4 działania:

- AddRoundKey (XOR z podkluczem rundy),
- SubBytes (podstawianie bajtów),
- ShiftRows (mieszanie bajtów, permutacja bajtów),
- MixColumns (transformacja liniowa).

Operacje te są odwracalne, zatem deszyfrowanie polega na wykonaniu rund w odwrotnej kolejności. Do dziś nie ma praktycznych ataków na szyfr AES.

MATERIAŁY DODATKOWE

- Siła szyfru nie polega na jego ukryciu, na utajnieniu sposobu działania, okazało się, że prędzej czy później sposób działania szyfru można odkryć, przyjęto, że jedynym tajnym parametrem szyfrów będzie klucz k.
- **Szyfr DES** (**Data Encryption Standard**)
  - Pierwszy standard szyfrowania symetrycznego
  - Rzeczywista długość — 56 bitów
  - Triple DES — potrójne szyfrowanie trzema różnymi kluczami uznaje się za bezpieczne, samodzielny DES jest łatwy do złamania atakiem brutalnym

SZYFROWANIE ASYMETRYCZNE

W przypadku szyfrów symetrycznych, rozmiar bloku i rozmiar klucza gwarantuje nam szyhbkość działania i praktyczne bezpieczeństwo. Jednak w przypadku protokołu wymiany informacji w celu zapewnienia poufności informacji, musimy zapewnić by klucz tajny był przekazany przez nadawcę do odbiorcy w bezpieczny sposób. Nie można klucza przesłać niezabezpieczonym kanałem, tekstem jawnym. Nie można go zaszyfrować szyframi symetrycznymi, bo wymagałoby to kolejnego klucza symetrycznego, którego również nie można przekazać. Rozwiązaniem tego problemu są protokoły wymiany klucza (przykładowo protokół Diffie-Hellmanna) lub zastosowanie szyfrów asymetrycznych.

LICZBY PIERWSZE

Szyfry asymetryczne mają u swojej podstawy problemy trudne obliczeniowo, algorytmy których czas działania nie jest wielomianowy, tylko przynajmniej wykładniczy. Jednym z takich problemów jest **faktoryzacja**, czyli rozkład liczby na czynniki pierwsze. W przypadku gdy $n = p * q$, dla wystarczająco dużych p i q, znając n wyznaczenie pary p, q jest problemem trudnym obliczeniowo.

Naturalnym rozwiązaniem dla losowania dużych liczb pierwszych jest losowanie b bitowego ciągu (z 1 na początku) i sprawdzenie czy dostaliśmy liczbę pierwszą. Powtarzamy losowanie t razy lub dopóki nie uzyskamy liczby pierwszej. **Postulat Bertranda** — dla dowolnego $b > 1$ liczby pierwsze długości b stanowią co najmniej $1/3b$ liczb naturalnych długości b. Co oznacza że otrzymamy liczbę pierwszą w czasie wielomianowym względem b.

Sprawdzenie czy liczba jest pierwsza jest zasobożerne, dlatego istnieje sposób aby nie robić pełnego rozkładu na czynniki pierwsze — **algorytm AKS** — deterministyczny algorytm wielomianowy z pierwotną złożonością czasową oszacowaną na $O(\log(n)^{12})$; aktualnie poprawiona wersja algorytmu umożliwia zredukowanie czasu do $O(\log(n)^6)$. **W praktyce wykorzystujemy szybsze algorytmy probabilistyczne**.

Test Millera-Rabina

- Test probabilistyczny — może się pomylić z zaniedbywalnym prawdopodobieństwem
- Odpowiada tylko TAK lub NIE
- W przypadku gdy p jest liczbą pierwszą test Millera-Rabina zawsze odpowie TAK.
- W przypadku gdy p jest liczbą złożoną (nie jest liczbą pierwszą) test Millera-Rabina odpowiada NIE z zaniedbywalnym prawdopodobieństwem. (false positive)
- W wersji uproszczonej możemy zapisać test Millera-Rabina w następującej postaci:
  - Wejście: liczba n oraz parametr t
  - Wyjście: liczba n jest pierwsza lub złożona

1. for i = 1 to t:
2. wybierz losowo $a \in \{1, 2, \ldots, n-1\}$ (świadek pierwszości);
3. jeśli $a^{n-1} \not\equiv 1 \pmod{n}$, zwróć "złożona";
4. w przeciwnym razie zwróć "pierwsza".

Jeżeli n jest liczbą pierwszą, algorytm zawsze zwróci "pierwsza", ponieważ dla każdego $a \in \{1, 2, \ldots, n-1\}$ zachodzi $a^{n-1} = a^{\varphi(n)} \equiv 1 \pmod{n}$ na mocy twierdzenia Eulera.

Jeżeli n jest liczbą złożoną, algorytm odpowie "złożona" z prawdopodobieństwem co najmniej $1 - 2^{-t}$, **z wyjątkiem liczb Carmichaela**.

- W pełnej wersji testu Millera-Rabina warunek jest bardziej rozbudowanyu
- Dodatkowo możemy od razu odrzucać liczby parzyste oraz potęgi
- Prawdopodbieństwo, że test nie zwróci "złożona" w żadnej iteracji wynosi co najwyżej 2^{-t}
- Złożoność algorytmu jest wielomianowa ze względu na liczbę bitów n oraz t

Szyfrowanie asymetryczne

Analogiczna definicja do szyfrowania symetrycznego, tym razem jednak będziemy używać pary kluczy zamiast jednego klucza tajnego $k \in \mathcal{K}$.

1. generowania klucza (algorytm probabilistyczny Gen), który generuje parę kluczy: klucz prywatny k_{priv} oraz klucz publiczny k_{pub};
2. szyfrowania (algorytm $\mathsf{Enc}_{k_{\mathsf{pub}}}(m)$);
3. deszyfrowania (algorytm $\mathsf{Dec}_{k_{\mathsf{priv}}}(c)$).

Kryptosystem asymetryczny musi spełniać następującą własność: dla każdego m zachodzi

$$\mathsf{Dec}_{k_{\mathsf{priv}}}\big(\mathsf{Enc}_{k_{\mathsf{pub}}}(m)\big) = m.$$

(w uproszczeniu wiadomość jest równa wiadomości zaszyfrowanej kluczem publicznym i odszyfrowana kluczem prywatnym)

Szyfr RSA

- Korzysta z dwóch losowo wybranych liczb pierwszych p i q długości b bitów
- Do szyfrowania i deszyfrowania wykorzystuje szybki algorytm potęgowania modularnego (fastexp)
- **Bezpieczeństwo**
  - Atakujący, mając moduł n, nie może wyznaczyć $p' > 1$ i $q' > 1$ takiego, że $n = p'q'$. Nie istnieje znany wielomianowy algorytm faktoryzacji. Dla odpowiednio dużego n nie można go rozłożyć na czynniki pierwsze.
  - Dziś n powinno być rzędu 2^{2048}, dla dłuższego zabezpieczenia 2^{4096}. Są to zatem liczby 2048- lub 4096-bitowe.
  - Rozmiar minimum 2048 bitów klucza można używać do roku 2030.
  - Czysta wersja RSA nie jest odporna na atak z wybranym tekstem jawnym, przykładowo posiadając klucz publiczny dla krótkich wiadomości $m < B$, atakujący może sprawdzić wszystkie wiadomości $m < B$ szyfrując je kluczem publicznym i weryfukując $c = \mathsf{Enc}_{k_{\mathsf{pub}}}(m)$.
  - Wersja ta nie jest też odporna na atak z wybranym szyfrogramem (CCA). Przykład:
    1. Atakujący przechwytuje szyfrogram $c = \mathsf{Enc}_{k_{\mathsf{pub}}}(m)$.
    2. Atakujący oblicza szyfrogram c' tajnej wiadomości $2m$ w następujący sposób $c' = c \cdot 2^e \bmod n = m^e \cdot 2^e \bmod n = (2m)^e \bmod n$.
    3. Atakujący prosi o zdeszyfrowanie $c' = 2^e c \bmod n$ i otrzymuje $2m$.
    4. Dzieli wynik przez 2 i odzyskuje wiadomość m.
  - Niebezpieczne jest również stosowanie małych wartości wykładnika e.
  - Poprawa sytuacji:
    - Wykorzystanie **RSA-OAEM** (**RSA Optimal Asymmetric Encryption Padding**, wykorzystanie funkcji hasujących oraz dodatkowego ciągu losowego
    - Wykorzystanie RSA do wymiany klucza symetrycznego wraz z uwierzytelnieniem **RSA-KEM** (**RSA Key Encapsulated Mechanism**)

Inne kryptosystemy asymetryczne

- Problem logarytmu dyskretnego — kryptosystem ElGamala
- Wykorzystanie krzywych eliptycznych (Elliptic Curves)
- Wykorzystanie teorii krat (Lattice-based cryptography) wraz z KEM:
  - kryptosystem NTRU
  - kryptosystem Crystal-Kyber
  - szyfry z grupy Post-Quantum Cryptography, odporne na ataki z wykorzystaniem komputerów kwantowych

Zastosowania

- Wymiana klucza symetrycznego
- Uwierzytelnianie
- Podpis elektroniczny
- Szyfry homomorficzne — szyfrowanie, które pozwala na operowanie na zaszyfrowanym dokumencie bez jego deszyfrowania

Protokół Diffie-Hellmanna

- Protokół generowania klucza symetrycznego przez dwie osoby
- Opublikowany w 1976r.
- Wykorzystywany w wielu programach i standardach bezpiecznej komunikacji, przykładowo GnuPG, GPG, protokołach VPN, SSH, TLS.
- Ze względu na długość życia klucza prywatnego wyróżniamy 2 wersje protokołu:
  - **statyczną** — wykorzystuje długoterminowe klucze prywatne, które nie zmieniają się po zakończeniu połączenia
  - **efemeryczną** (DHE) — generuje klucze prywatne dla każdego nowego połączenia
- U podstaw leży trudność problemu logarytmu dyskretnego. Daje nam on bezpieczeństwo w razie podsłuchania przesyłanych danych, znając wartości publiczne oraz przesyłane ukryte sekrety Ewa nie może wyznaczyć wspólnego klucza ani tajnych sekretów. Protokół jednak jest podatny na atak aktywny w trakcie którego nie tylko podsłuchiwane są dane, ale atakujący może ingerować w przesyłane informacje. Można go zaatakować atakiem **Man-in-the-middle**, atakujący może podmienić publiczne parametry przesyłane pomiędzy uczestnikami na własne i ustalić dwa osobne klucze symetryczne z każdym użytkownikiem, a następnie odszyfrowywać przesyłane pomiędzy nimi zaszyfrowane wiadomości. Ochroną przeciw takiemu atakowi jest dodanie uwierzytelnienia stron protokołu.

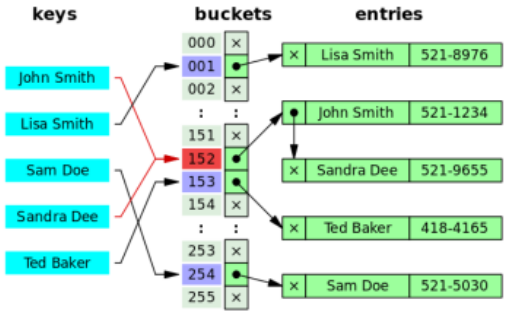
Funkcje haszujące

Wprowadzenie

Funkcje haszujące to kolejny obok szyfrowania symetrycznego i asymetrycznego element budujący bezpieczeństwo systemów komputerowych. Szyfrowanie do tej pory stosowaliśmy w celu zapewnienia poufności danych, wymiany klucza do komunikacji. **Funkcje haszujące stosowane są podczas uwierzytelniania użytkowników, uwierzytelniania dokumentów w trakcie podpisów elektronicznych czy zapewnienia integralności danych.**

Zadaniem funkcji haszującej jest zamiana tekstu dowolnej długości $m \in \{0,1\}^*$**, do ciągu bitowego o ustalonej długości**, zazwyczaj krótszej od długości wiadomości. Stąd czasem nazywamy funkcje hasujące **funkcjami skrótu**. Głównym zadaniem bezpiecznych funkcji haszujących jest **unikanie kolizji**, czyli dwóch róznych tekstów, które mają taki sam skrót. Funkcje haszujące wykorzystują techniki znane z szyfrów symetrycznych (**konstrukcja funkcji**) oraz mają wiele zastosowań w szyfrach asymetrycznych (**podpis elektroniczny**)

Funkcje haszujące są również stosowane w algorytmach czy bazach danych do szybkiego wyszukiwania elementów. Skrót jest wówczas indeksem w tablicy w której przechowywujemy dane. Przykładowo element x jest przechowywany pod indeksem $H(x)$. Do pobrania elementu wystarczy czas $O(1)$ plus ewentualnie przejrzenie listy elementów w danym indeksie ze względu na możliwe kolizje. Dobra funkcja hashująca będzie mieć niewiele kolizji, ponieważ wydłużają czas dostępu do danych.



Funkcje hashujące wykorzystywane w bezpieczeństwie muszą być bezkolizyjne, na poziomie praktycznym, gdyż z samej definicji funkcji hashującej kolizji nie da się uniknąć. W przypadku struktur danych, dane przechowywane są niezależnie od funkcji hashującej, w bezpieczeństwie atakujący będzie wybierał tak dane aby doprowadzić do kolizji. (np.: inny dokument ma tą samą długość bajtów, możliwa kolizja w checksumie) Bezpieczne funkcje haszujące są o wiele trudniejsze w konstrukcji niż funkcje hashujące w strukturach danych (mapy). Funkcja haszująca H powinna zatem być odporna dla każdego wielomianowego algorytmu probabilistycznego, którego celem jest znalezienie kolizji — bezpieczeństwo obliczeniowe. Ponieważ wynik działania funkcji jest krótszy od tekstu wejściowego, kolizja musi występować, ale powinna być trudna do znalezienia.

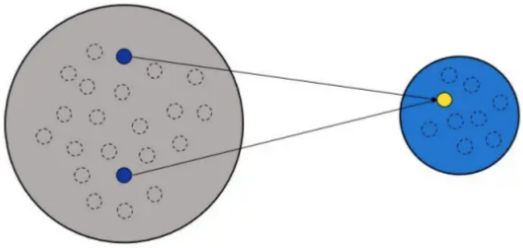


Figure 1: Kolizja funkcji $y = H(x)$

Funkcje haszujące mogą być zdefiniowane jako funkcje z kluczem albo funkcje bez klucza. Funkcje z kluczem mają dwie dane na wejściu, klucz s oraz tekst (ciąg bitów) x.
Definicja. Funkcja haszująca o wyjściu długości l jest parą algorytmów probabilistycznych o czasie wielomianowym (PPT), oznaczanych (Gen, H), zdefiniowanych następująco:

1. Gen jest algorytmem PPT, który na wejściu otrzymuje parametr bezpieczeństwa 1^n, ustala długość klucza na n i zwraca klucz $s \in \{0,1\}^n$;

2. H na wejściu otrzymuje klucz s oraz ciąg $x \in \{0,1\}^*$ i zwraca ciąg $H_s(x)r \in \{0,1\}^{l(n)}$, gdzie n jest długością klucza s.

Dla uproszczenia będziemy czasami nazywać funkcją haszującą samo H_s.
Dodatkowo wymagamy, by dla H_s oraz losowo wybranego klucza s, znalezienie kolizji dla H_s było bardzo trudne. **Klucz s nie musi być tajny**, kolizje muszą być trudne do znalezienia nawet dla atakującego znającego s. Jeżeli definiujemy H_s dla $x \in \{0,1\}^{l'(n)}$ takich, że $l'(n) > l(n)$, wówczas para (Gen, H) jest funkcją haszującą o ustalonej długości wejścia l' i będziemy ją nazywać **funkcją kompresji**. (l prim — długość bloku kompresowanego, l — długość wyjściowa po kompresji)

BEZPIECZEŃSTWO

Bezpieczeństwo funkcji haszującej możemy zdefiniować w postaci eksperymentu poszukiwania kolizji $\mathsf{HashColl}_{A,\Pi}(n)$, dla funkcji haszującej $\Pi = (\mathsf{Gen}, \mathsf{H})$, atakującego A oraz parametru bezpieczeństwa n.

1. Klucz s jest generowany algorytmem $\mathsf{Gen}(1^n)$.
2. Atakujący A otrzymuje s oraz wybiera dwa wejścia x, x' (jeżeli Π ma wejścia długości $l'(n)$, wówczas $x, x' \in \{0,1\}^{l'(n)}$).
3. Wynikiem eksperymentu jest 1 wtedy i tylko wtedy, gdy $x \neq x'$ oraz $H_s(x) = H_s(x')$, co oznacza, że atakujący znalazł kolizję.

Definicja. Funkcja haszująca $\Pi = (\mathsf{Gen}, \mathsf{H})$ jest **bezkolizyjna** (**collision resistant**), jeżeli dla wszystkich atakujących PPT A istnieje funkcja zaniedbywalna negl taka, że

$$\Pr\left[\mathsf{HashColl}_{A,\Pi}(n) = 1\right] \leq \mathsf{negl}(n).$$

W praktyce często stosujemy funkcje haszujące bez klucza o ustalonej długości wyniku, czyli funkcje $H : \{0,1\}^* \to \{0,1\}^l$. Są to funkcje bezkolizyjne w praktycze — znalezienie kolizji jest trudne obliczeniowo. W wielu sytuacjach zamiast wymogu bezkolizyjności stosujemy dwa inne wymogi dotyczące bezpieczeństwa. Funkcje haszującą nazywamy **silnie bezkonfliktową** (**second-preimage resistance**) gdy dla danego klucza s nie jest praktycznie możliwe dla PPT atakującego znalezienie pary tekstów $x \neq x'$ takiego, że $H_s(x) = H_s(x')$. (podmiana dokumentu jest niemożliwa) Funkcje haszującą nazywamy **słabo bezkonfliktową** (**preimage resistance**) gdy dla danego klucza s nie jest praktycznie możliwe dla PPT atakującego znalezienie pary tekstów $x' \neq x$ takiego, że $H_s(x') = H_s(x)$. (preimage attack — znalezienie tekstu do zadanego skrótu; first-preimage attack — mając dany skrót znajdź wiadomość; second-preimage attack — mając ustaloną wiadomość znajdź drugą wiadomość) Generalnie jest to to samo co **jednokierunkowość** (**one-way**). (hasza nie da się odwrócić, mając sam hash hasła nic nie można z nim zrobić) W praktyce konskrukcja funkcji haszującej opiera się na bezkolizyjnej funkcji kompresji o określonej długości wejścia.

BUDOWA

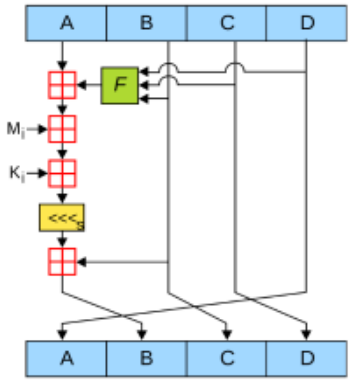
Schemat Merkle-Damgarda procesuje długie wiadomości blok za blokiem, wkładając wyjście jednej kompresji w krok drugiej. Jeżeli funkcja kompresji dla jednego bloku jest bezkolizyjna to funkcja dla wszystkich bloków też jest bezkolizyjna.

KRYPTOANALIZA

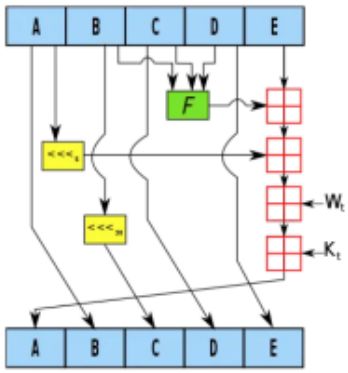
Jednym z ataków na funkcje haszujące jest **atak urodzinowy**, jest to uniwersalny atak na każdą funkcję haszującą. Ustala on dolną granicę rozmiaru skrótu. **Paradok urodzinowy** — Jeżeli jest q osób w pokoju, jakie jest prawdopodobieństwo, że dwie osoby mają urodziny tego samego dnia? (jeżeli w pokoju są 23 osoby to prawdopodibeństwo jest większe od 1/2) Atak urodzinowy pozwala jedynie na znalezienie kolizji, co zazwyczaj nie jest przydatne, bardziej potrzebne jest znalezienie kolizji dwóch sensownych wiadomości. Wykorzystując atak urodzinowy możemy róœnież znaleźć takie kolizje. W łatwy sposób możemy zapisać tą samą wiadomość na wiele różnych sposobów, przykładowo dodając znaki białe na końcu tekstu, w środku tekstu, wymieniając słowa w wiadomości na synonimy. (w skrócie istnieje szansa i metoda na to że podpisany dokument może zostać podmieniony, jest na to bardzo mała szansa ale metody takie jak SHA-1 i MD5 są niewystarczające bo ich złamanie aktualnie zajmuje bardzo mało czasu)

Standardy

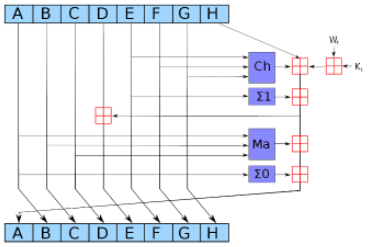
- **MD5** — 128 bitowy skrót. Złamana w 2004 roku pokazano metodę na znajdowanie kolizji.



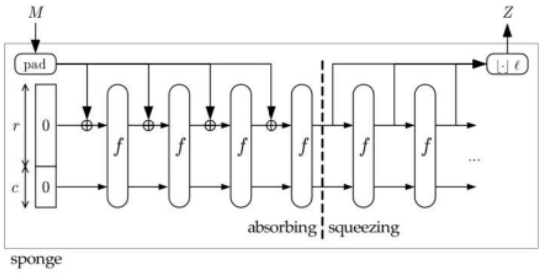
- **SHA-0**, **SHA-1** — rodzina standardowych funkcji haszujących SHA. SHA-1 (160 bitowy skrót), poprawiony następca SHA-0. Obydwie metody złamane, można go złamać dużo szybciej niż atakiem urodzinowym.



- **SHA-2** — aktualny standard haszowania, w praktyce używane SHA-256 i SHA-512, konstrukcja podobna do SHA-1



- **SHA-3** — nowa standardowa funkcja haszująca, Keccak. Struktura całkowicie odmienna od SHA, konstrukcja gąbki. Długość skrótu 256, 384, 512 bitów.



ZASTOSOWANIA

"Odcisk palca" (**fingerprint**) — stosując bezkolizyjną funkcję haszującą H na pliku, skrót pliku staje się unikalnym identyfikatorem tego pliku. Jeżeli inny plik ma ten sam identyfikator, oznacza to kolizję w funkcji H, co nie powinno być praktycznie możliwe. Zatem skrót $H(x)$ pliku x, staje się odciskiem palca, co pozwala zwefyfikować, czy dwa pliki są takie same za pomocą porównania skrótów. **Przykładowe zastosowania odcisków palcy**:

- **Poszukiwanie wirusów** — stosuje się bazy danych przechowujące skróty znanych wirusów, a następnie porównuje się ze skrótami pobranych plików, załączników. Przechowujemy jedynie odciski zawirusowanych plików.
- **Eliminowanie duplikatów** — przykładowo w chmurach danych przechowywane są te same pliki przez wielu użytkowników (np. filmy), wówczas wystarczy przechować tylko jeden plik i dać dostęp dla wielu użytkowników. Wraz z plikiem przechowujemy ich skróty.
- **Technologia Blockchain** — przechowuje cyfrowy odcisk palca dokumentu lub zbioru danych. Weryfikuje czy nastąpiły zmiany w plikach.
- **Peer-to-peer (P2P)** — system współdzielenia plików, przechowyjemy tablice (Distributed Hash Tables, DHT) na serwerach w celu wyszukiwania plików. Tabele przechowują skróty dostępnych plików, które są identyfikatorami plików, zajmują bardzo mało miejsca.
- **Przechowywanie haseł** — jedno z najważniejszych i najczęstszych zastosowań funkcji haszujących.
- **Tworzenie kluczy** (**Key Derivation**) — w szyfrowaniu symetrycznym potrzebujemy losowych kluczy o rozkładzie jednostajnym (równo prawdopodobnych). W praktyce są one generowane na podstawie wspólnego sekretu, hasła, danych biometrycznych, nie są to jednak dane o rozkładzie jednostajnym. Klucz symetryczny musi być ustalonej długości, posiadane sekrety lub hasła mogą mieć inną długość. Skracanie lub wydłużanie tych danych również zaburza jednostajność.
  - **Funkcja HMAC** — połączenie MAC (Message Authentication Code) z funkcją haszującą z kluczem, w celu uwierzytelniania i integralności przesyłanych treści.

MATERIAŁY DODATKOWE

- **Zastosowania funkcji hashujących**
  - Podpis elektroniczny
  - Testowanie integralności danych (MDC)
  - FUnkcje haszujące z kluczem (MAC)
  - Przechowywanie haseł w systemach komputerowych
  - Generatory pseudolosowe
  - Generowanie kluczy sesyjnych
  - Znakowanie czasem
  - Kryptowaluty i technologia blockchain
- `https://medium.com/@short_sparrow/how-hmac-works-step-by-step-explanation-with-examples-f4aff5efb4`
  - **HMAC** — podpis elektroniczny z kluczem prywatnym (szyfrowanie symetryczne) (często wykorzystywana metoda do JWT)
  - **RSASSA** — podpis elektroniczny z kluczem publicznym i prywatnym

Podpis elektroniczny

Wprowadzenie

W przypadku protokołu wymiany tajnej wiadomości potrzebujemy:

- Klucza symetrycznego dla szyfrowania wiadomości
- Klucza publicznego i prywatnego do wymiany klucza symetrycznego
- Ewentualnie publicznych parametrów do protokołu ustalenia klucza symetrycznego
- Uwierzytelnienia stron protokołu ze względu na taki aktywne, podszywanie się pod strony protokołu

W celu uwierzytelnienia stron protokołu będą nam potrzebne podpisy elektroniczne oraz certyfikacja klucza publicznego. W praktyce stoswanie podpisów elektronicznych wymaga również użycia bezpiecznych funkcji haszujących. Podpis elektroniczny **poświadcza prawdziwość dokumentu oraz potwierdza osobę składającą podpis pod dokumentem**. W odróżnieniu podpis odręczny poświadcza jedynie osobę, gdyż składany może być pod dowolnym dokumentem i powinien zawsze wyglądać bardzo podobnie. Podpis odręczny nie zależy od treści dokumentu, może być nawet złożony na pustej kartce. Co najwyżej jest związany z nośnikiem dokumentu. Podpis odręczny jest wyuczony, nie może być złożony idealnie przez inną osobę, ale łatwo go skopiować. Podczas weryfikacji porównujemy go z wzorem dokumentów, mogą wystąpić problemy z ponownym złożeniem identycznego podpisu przez tą samą osobę. Jak widać jest to rozwiązanie problematyczne, niepewne, trudne do zweryfikowania, a jeżeli ma być zrobione wiarygodnie, to będzie wolne i drogie (grafolog).

Wersja elektroniczna wymaga:

- urządzenia do składania podpisu (ewentualnie aplikacji zależnie od poziomu bezpieczeństwa)
- tajnych danych służących do złożenia podpisu, dostępnych jedynie dla osoby składającej podpis
- publicznych danych wiarygodnie przypisanych do osoby składającej podpis

Podobnie jak w szyfrach asymetrycznych wymagamy ustalenia trzech protokołów:

- generowanie kluczy
- składanie podpisu elektronicznego
- weryfikacja podpisu elektronicznego

W podpisie elektronicznym podobnie jak w szyfrach asymetrycznych korzystamy z pary kluczy (publiczny i prywatny) Klucz **prywatny służy do podpisywania dokumentu**, tylko osoba znająca klucz prywatny może złożyć podpis. **Weryfikacja podpisu odbywa się za pomocą klucza publicznego**, który jest powiązany z kluczem prywatnym. Nie można wyznaczyć klucza prywatnego z publicznego lub złożonych podpisów. Kluczy publiczne muszą być wiarygodnie powiązane z właścicielem (w praktyce zazwyczaj w formie certyfikatu). **Złożony podpis zależy od klucza prywatnego podpisującego oraz od treści podpisywanej wiadomości**. Podpis jest ciągiem bitów dodawanych do wiadomości. Nie można go skopiować do innej wiadomości. **Podpisywanie i weryfikacja podpisu nie są tym samym co szyfrowanie i deszyfrowanie**, chociaż używamy klucza publicznego i prywatnego. Ze względów bezpieczeństwa para kluczy do podpisywania powinna być inna niż para kluczy do szyfrowania. Jest to kolejne działanie służące do uzyskania bezpieczeństwa systemów, obok szyfrowania symetrycznego, szyfrowania asymetrycznego i haszowania.

Bezpieczeństwo

Dla ustalonego klucza publicznego k_pub strony podpisującej (signer), podrobienie podpisu (forgery) oznacza znalezienie innej wiadomości m wraz z pasującym do niej podpisem s, taką która nie została wcześniej podpisana. Atakujący nie może wyznaczyć podpisu wiadomości nawet znając podpisy wielu różnych wiadomości, nawet wybranych przez siebie. Formalnie możemy zapisać to jako eksperyment atakującego A oraz wielkości n, parametru bezpieczeństwa (długość klucza).
Schemat podpisu nazywamy odpornym na **adaptacyjny atak z wybraną wiadomością** (**adaptive chosen-message attack**) lub po prostu bezpiecznym, jeżeli dla wszystkich PPT atakujących A istnieje funkcja zaniedbywalna $negl$ taka, że

$$\Pr\left[\mathsf{SigForge}_{A,\Pi}(n) = 1\right] \leq \mathsf{negl}(n).$$

(szansa atakującego na podrobienie podpisu jest zaniedbywalna)

Wyróżniamy generalnie dwa rodzaje podpisów:

- **z wykorzystaniem funkcji haszujących** — zamiast całej wiadomości podpisywany jest jej skrót, podpis publikowany jako para
- **bez wykorzystania funkcji haszujących** — dla bardzo krótkich wiadomości, podpisujemy wiadomosc i publikujemy wyłącznie jej podpis s. Podczas weryfikacji z s wyznaczana jest oryginalna wiadomość oraz potwierdzenie poprawności podpisu.

Podpis RSA

Podpis RSA jest przykładem bezpiecznego podpisu elektronicznego opierający się na szyfrze asymetrycznym RSA.

Czysta wersja podpisu RSA nie jest bezpieczna, pomimo trudnośći wyznaczenia klucza prywatnego, można używając klucza publicznego "odwrócić" działanie weryfikacji podpisu. Mając klucz publiczny wybieramy dowolne $s \in \mathbb{Z}_n^*$, obliczamy $m = s^e \bmod n$ na podstawie klucza publicznego i mamy podrobiony podis (m, s) bez udziału właściciela klucza prywatnego. Jak mamy szczęście to m będzie podobny do jakiegoś tekstu. Małe zastosowanie praktyczne, jednak nie zgadza się z poprzednimi definicjami bezpieczeństwa.
Drugi atak jest trochę bardziej praktyczny. Jeżeli atakujący może uzyskać dwa podpisy, wówczas atakujący może uzyskać podpis pod inną wiadomością dla klucza publicznego. Wybiera w tym celu dwie wiadomości. Dostaje ich podpisy i oblicza $s = s_1 * s_2 \bmod n$, która jest podpisem pod wiadomością. (uproszczenie bez matematyki)

Zabezpieczeniem podpisu RSA jest dodanie funkcji haszującej w podpisie, zamiast podpisywać wiadomości podpisujemy ich skróty. Dla ochrony przed dwoma wcześniejszymi atakami H powinna być nieodwracalna oraz trudno znaleźć m, m_1, m_2 takie, że $H(m) = H(m_1) * H(m_2)$, dodatkowo musi być bezkolizyjna — taką odmianę RSA nazywamy **RSA-FDH** (**RSA Full-Domain Hash**, funkcja H jest funkcją losową, rozrzuca jednostajnie teksty po zbiorze $\mathbb{Z}_n^*$

Inne schematy podpisów:

- Oparte o problem logarytmu dyskretnego
  - podpis El-Gamala
  - podpis Schnorra
  - podpis DSA (Digital Signature Algorithm)
- Oparte o krzywe eliptyczne
  - podpis ECSDSA (Elliptic Curves Digital Signature Algorithm)

Podsumowanie

Dodając do podpisu elektronicznego wiarygodny sposób wymiany kluczy publicznych (PKI — Public Key Infrastructure), zrealizowany przykładowo za pomocą certyfikatów klucza publicznego wystawianych przez urzędy certyfikujące (CA — Certificate Authorities), rozwiązujemy problem aktywnego ataku Man-in-the-Middle. W praktyce realizacją protokołu bepzieczenego połączenia między dwoma stronami (klientem i serwerem) jest protokół TLS (Transfer Layer Security), który zapewnia poufność i integralność transmisji danych, a także uwierzytelnienia serwera i klienta.

W protokole tym wykorzystujemy wszystkie poznane techniki:

- W trakcie fazy nawiązywania połączenia (**Handshake Protocol**) wykorzystywane są certyfikaty do wymiany klucza publicznego serwera (ewentualnie również klienta), liczby losowe do wygenerowania klucza sesyjnego,
- Certyfikaty są podpisane elektronicznie przez zaufaną trzecią stronę (CA). Uwierzytelniamy uczestników protokołu (głównie serwer).
- Ustalane są sposoby szyfrowania, haszowania, wymiany kluczy sesyjnych (**Cipher Suites**).
- Po ustaleniu sposobu szyfrowania wiadomości i ustaleniu klucza sesyjnego, dane szyfrowane są szyfrem symetrycznym (blokowym lub strumieniowym, w ustalonych trybach pracy) (**Record-Layer Protocol**). Tu może też uwierzytelniać się klient (np. przesyłając zaszyfrowane hasło)

MATERIAŁY DODATKOWE

Podpis elektroniczny:

- identyfikuje osobę podpisującą
- może być złożony jedynie przez osobę podaną jako osoba podpisująca
- jest związany z podpisanym tekstem w ten sposób, że nie pasuje jako podpis do żadnego innego tekstu

CERTYFIKATY

POJĘCIE CERTYFIKATU

Poprzez uwierzytelniania na stronach WWW użytkownik podaje hasło uzgodnione z serwerem. Niestety szybko każdy serwer posiada wiele róznych haseł od użytkowników, ich przechowywanie jest kłopotliwe. Podobna sytuacja ma miejsce podczas komunikacji między serwerami. Zamiast porowadzić weryfikację na serwerze, zakłada się, że osoby i maszyny uczestniczące w komunikacji posiadają **certyfikaty**. Zapewniają one godną zaufania weryfikację tożsamości. Certyfikaty tworzone są w sposób hierarchiczny.

Certyfikat (certyfikat klucza publicznego) — elektroniczne zaświadczenie, wystawione przez podmiot świadczący usługi certyfikacyjne (centrum certyfikacji), które zawiera dane służące do weryfikacji podpisu elektronicznego.

Certyfikat może być przypisany zarówno do osoby fizycznej jak i domeny, adresu IP lub urządzenia sieciowego itd. Certyfikat zawiera m.in.:

- identyfikator wystawcy certyfikatu
- identyfikator użytkownika
- klucz publiczny użytkownika
- okres ważności
- numer seryjny certyfikatu

Certyfikat jest podpisany przez wystawcę

Rodzaje certyfikatów

- **Certyfikat kwalifikowany** — spełnia wymaga Ustawy o podpisie elektronicznym, wydany przez kwalifikowany podmiot świadczący usługi certyfikacyjne. Podpis elektroniczny weryfikowany za pomocą certyfikatu kwalifikowanego oraz złożone za pomocą bepziecznego urządzenia do składania podpisu elektronicznego, jest równoważny podpisowi własnoręcznemu. Wymaga ustawy i przepisów wykonawczych dotyczących m.in. poziomu zabezpieczeń sprzętu, niepowtarzalności pewnych danych i metod obsługi klientów. Certyfikat kwalifikowany może być wydany **jedynie osobie fizycznej**.
- **Certyfikat niekwalifikowany** (zwykły, komercyjny) — służy do opatrywania dokumentów podpisem niekwalifikowanym (zwykłym) oraz m.in. do szyfrowania poczty elektronicznej i plików, autoryzacji w systemach IT itp.
- **Certyfikat niekwalifikowany zaufany** — wydawany zgodnie ze standardem **WebTrust**. Zgodność ta oznacza, że certyfikaty niekwalifikowane są uznawane za zaufane globalnie we wszystkich rozpowszechnionych przeglądarkach internetowych na świecie oraz we wszystkich produktach Microsoft Corp.
- **Certyfikat Atrybutów (AC)** — elektroniczne zaświadczenie, za pomocą którego wartości atrybutów są przyporządkowane do wskazanej w zaświadczeniu osoby i powiązane z danymi identyfikacyjnymi tej osoby.

Oczywiście fizycznie certyfikat jest ciągiem danych zapisanych na odpowiednim nośniku. Najpopularniejszym standardem certyfikatów jest **X.509 składający się z pól**:

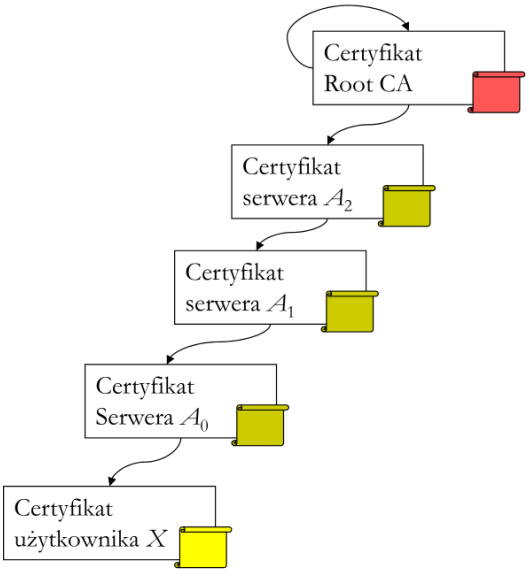
- wersja
- numer seryjny
- wydawca certyfikatu
- data ważności
- podmiot dla którego certyfikat został wystawiony
- klucz publiczny
- identyfikator algorytmu podpisu składanego pod treścią certyfikatu
- podpis organu wydającego certyfikat

Format certyfikatów i **list CRL** (unieważnionych certyfikatów) standardu X.509 opisany jest w RFC 2459 oraz jako standardu ITU-T Recommendation X.509. Treść certyfikatu składa się z:

- **pól podstawowych**, które muszą wystąpić w każdym certyfikacie
- **rozszerzeń**, czyli pól, które można lecz nie trzeba umieścić w certyfikacie, dzielą się na standardowe i niestandardowe (które każdy wystawca certyfikatów może utworzyć i zarejestrować)
  - rozszerzenia **krytyczne** zawierają informacje istotne dla ustalenia zaufania (bądź nie) do certyfikatu — nie powinno się akceptować certyfikatu w przypadku, gdy oprogramowanie i sprzęt nie potrafią zinterpretować treści rozszerzeń krytycznych
  - rozszerzenia **niekrytyczne** zawierają informacje mniej istotne z punktu widzenia bezpieczeństwa — możliwe jest wówczas zaakceptowanie certyfikatu w przypadku, gdy oprogramowanie i sprzęd nie rozpoznają takich rozszerzeń

Hierarchia

Serwer A_0 wystawia certyfikat użytkownikowi X, potwierdza go swoim podpisem elektronicznym. Do weryfikacji tego podpisu potrzebne są dodatkowe dane, które są zawarte w certyfikacie nadanym A_0 przez serwer A_1. Do uwierzytelnienia certyfikatu nadanego A_0 jest on podpisany przez A_1, ale jak zweryfikować ten podpis?
Do tego potrzebny jest certyfikat A_1 wystawiony i podpisany przez serwer A_2 itd. W ten sposób powstaje hierarchia, która zakończy się w **CA**. Oznacza to, że przekazujemy zawsze nie jeden certyfikat a listę certyfikatów (ścieżka certyfikacji). Pozwala to na zweryfikowanie certyfikatu użytkownika X bez angażowania serwerów A_0, A_1, A_2,..., oczywiście z wyjątkiem wystawienia samego certyfikatu przez serwer.
Na szczycie hierarchii są specjalne serwery zwane **Certificate Authority (CA)**. Każdy z nich jest mocno zabezpieczony i służy tylko do wydawania certyfikatów. CA nie musi potwierdzać wszystkich tworzonych certyfikatów bezpośrednio.



Status certyfikatu

Certyfikat może mieć jeden z trzech statusów:

- **certyfikat ważny** — jego termin ważności jeszcze nie upłynął, ponadto nie został unieważniony lub zawieszony
- **certyfikat unieważniony** — z określonych przyczyn stracił status ważności. Przyczynami utraty ważności mogą być ujawnienie klucza prywatnego odpowiadającego kluczowi publicznemu potwierdzonego certyfikatem, odkrycie luk bezpieczeństwa, jakie miały miejsce w procesie generowania tych kluczy, zgubienie przez właściciela klucza prywatnego, karty mikroprocesorowej zawierającej ten klucz, itp.
- **certyfikat zawieszony** — to certyfikat, co do którego istnieje podejrzenie, ze zainstniały przyczyny stanowiące podstawę unieważnienia i którego stosowanie zostało zawieszone

Zawieszenia i unieważnienia certyfikatu dokonuje centrum certyfikacji, które certyfikat wydało. Certyfikat raz unieważniony nie może stać się ponownie ważnym certyfikatem — inne rozwiązania mogłyby doprowadzić do paradoksów logicznych. Certyfikat zawieszony może zostać albo unieważniony albo z powrotem może stać się ważnym certyfikatem wskutek "odwieszenia". Okres zawieszenia zwykle jest z góry ograniczony (w Polsce certyfikat kwalifikowany może być zawieszony co najwyżej 7 dni)

Listy CRL

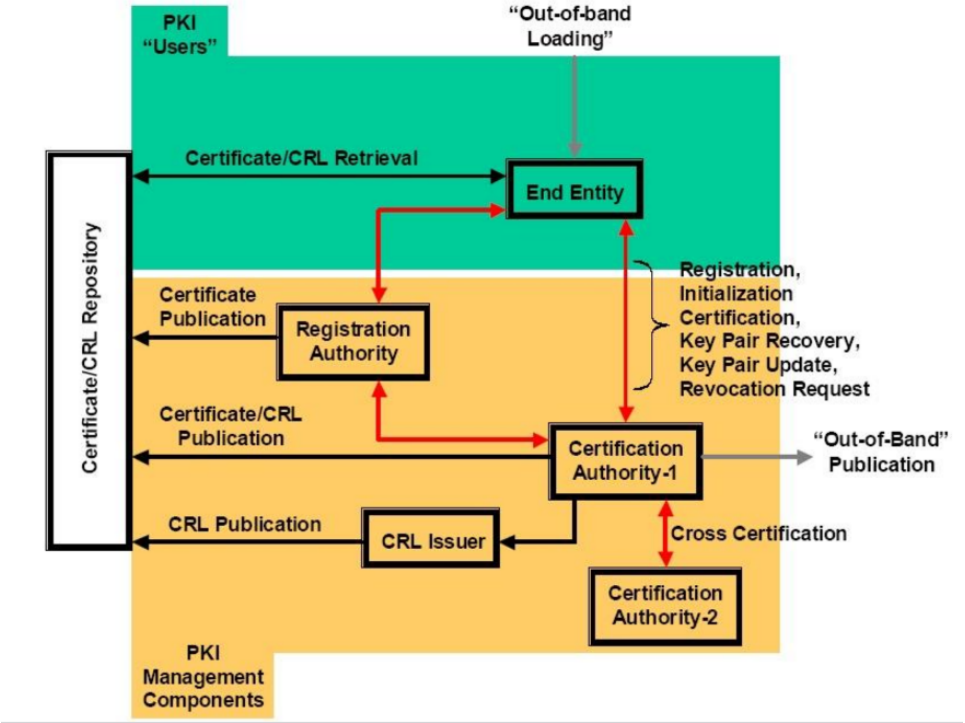
Lista CRL (**Certificate Revocation List**) jest podstawowym elementem mechanizmu sprawdzania statusu (ważności) certyfikatów. Podobnie do certyfikatów, dopuszcza się umieszczanie w listach CRL rozszerzeń. Podstawowe elementy występujące w każdej liście CRL to:

- **treść listy** — informacje ogólne dotyczące danej CRL oraz lista unieważnionych i zawieszonych certyfikatów
- **identyfikator algorytmu podpisu** składanego pod listą
- **podpis** pod treścią listy

Dla każdego certyfikatu na liście podawane są następujące informacje:

- numer seryjny certyfikatu
- data i godzina unieważnienia(/zawieszenia) certyfikatu
- wersja standardu X.509
- identyfikator algorytmu zastosowanego do podpisania treści listy
- nazwa wystawcy listy
- data i godzina wystawienia listy
- przewidywany czas wystawienia następnej CRL dotyczącej tej samej grupy certyfikatów
- lista unieważnionych certyfikatów (regułą jest uaktualnianie CRL po unieważnieniu lub zawieszeniu, dlatego do wystawienia nowej CRL może dojść wcześniej)

Model X.509



Składniki modelu:

- **użytkownik końcowy** (**End Entity**, **EE**) — ogólne pojęcie opisujące fizycznego użytkownika (osobę, firmę), urządzenie (serwer, router) lub dowolne inne pojęcie którego nazwa może pojawić się w polu certyfikatu klucza publicznego. Szczególnym przypadkiem może być to osoba fizyczna — certyfikat klucza publicznego do podpisu elektronicznego. Również RA z puunktu widzenia CA jest EE. EE musi dokonać procesu rejestracji przed wystawienie certyfikatu.
- **urząd certyfikacji** / centrum certyfikacji (**Certification Authority**, **CA**) wystawia certyfikaty, wystawiony certyfikat jest podpisany przez CA, może również wystawiać listy CRL (które mogą również być wystawiane przez inną instytucję). CA spełnia też role administracyjne: rejestrowanie podmiotów (z reguły jednak przez RA), tworzenie kopii zapasowych. Może również unieważniać certyfikaty, generować pary kluczy, przechowywać klucze, znakować klucze itp. Działalność centrum certyfikacji jest zwykle udokumentowana w standardowy sposób.
  Najważniejszymi dokumentami tego typu są:
  - **polityki certyfikacji** — powszechnie dostępne dokumenty określające sposób tworzenia, wydawania zasotosowania i unieważniania certyfikatów, obowiązki zarówno centrum certyfikacji jak i właścicieli certyfikatów, itp.
  - **polityki bezpieczeństwa** — dokumenty określające cały system środków bezpieczeństwa stosowanych w centrum certyfikacji, dokument ten nie jest publiczny, ale nie jest też ściśle tajny, gdyż bezpieczeństwo nie opiera się na poufności zastosowanych środków ostrożności
- **urząd rejestracji** (**Registration Authority**, **RA**) — opcjonalny składnik architektury, ma za zadanie potwierdzać dane użytkownika oraz dokonywać jego rejestracji, może spełniać dodatkowe funkcje:
  - sprawdzanie poprawności danych podmiotu składającego podanie o wystawienie certyfikatu (niekoniecznie osoby)
  - sprawdzenie czy podmiot posiada danyc klucz publiczny (proof of posession)
  - tworzenie części sekretu potrzbnego przy wystawianiu certyfikatu (współdzielenie poświadczenia)
  - generowania par klucz publiczny + prywatny na życzenie klienta
  - pośredniczenie między EE a CA
  - weryfikowanie parametrów klucza (liczby pierwsze, zakresy wartości itp.)

- **repozytorium** (**Repository**) — ogólne pojęcie dotyczące metody przechowywania certyfikatów oraz dokumentów CRL w sposób dostępny dla każdego użytkownika. Może być oparty o usługę LDAP (Lightweight Directory Access Protocol) lub po prostu przesyłanie plików protokołem FTP czy HTTP.
- **CRL** (**Certificate Revocation List**) — lista unieważnionych certyfikatów, ale nie przeterminowanych
- **wystawca CRL** (**CRL Issuer**) — opcjonalny składnik architektury, służy do wystawiania list CRL. Zazwyczaj wystawcą CRL jest po prostu CA, można jednak rozdzielić funkcję publikowania certyfikatów oraz list CRL.

Opis zaleności pomiędzy składowymi

- **Rejestracja**
  Każdy użytkownik końcowy (EE) musi najpierw dokonać rejestracji, aby móc przeprowadzić proces certyfikacji. Może być przeprowadzana on-line lub off-line (również może być mieszana). Ma za zadanie sprawdzenie danych osobowych użytkownika, często polega na przydzieleniu części sekretu, któ©y będzie potrzebny w dalszych etapach.
- **Inicjalizacja**
  Ma za zadanie rozpocząc powiązanie użytkownika z centrum certyfikacji. Użytkownik może na tym etapie otrzymać polityki bezpieczeństwa CA. Na tym etapie zazwyczaj generowana jestr para kluczy użytkownika (przez EE, CA, RA, itp.)
- **Certyfikacja**
  Użytkownik końcowy otrzymuje od centrum certyfikacji certyfikat. Jeżeli klucz publiczny EE jest generowany niezależnie, to należy róœnież przedstawić dowód poprawności. Po wystawieniu certyfikat jest wysyłany do użytkownika oraz do repozytorium.

Trzy powyższe procesy mogą być połączone w jeden w jednym centrum certyfikacji.

- **Odzyskiwanie klucza**
  W sytuacji gdy klucz do deszyfrowania (prywatny) zostanie zagubiony, ale jest przechowywany w centrum (zaufanym), można go odzyskać (Key Pair Recovery). Zazwyczaj zagubienie ma miejsce gdy zapomnimy hasłą lub PINu, wystąpi awaria dysku, sprzętu, karty elektronicznej. Sytuacja taka może również mieć miejsce gdy firma zwalnia pracownika, ale chce mieć dostęp do jego danych na komputerze firmowym. Odzyskiwanie kluczy podpisów elektronicznych jest niepraktykowane (wręcz zabronione w politykach CA)
- **Modyfikacja klucza**
  Certyfikaty są wystawiane na określony czas (np. 2 lata), gdy ten okres mija, należy wystawić nowy certyfikat dla nowego klucza (Key Pair Update). Ponowne wystawienie certyfikatu może również wystąpić w przypadku unieważnienia certyfikatu. Proces ten polega na wystawieniu nowej pary kluczy oraz nowego certyfikatu klucza publicznego. Osobnym procesem jest wystawienie nowego certyfikatu dla tej samej pary kluczy (Certificate Update)
- **Żądanie unieważnienia**
  Zdarzają się sytuacje, że przed wygaśnięciem certyfikatu klucz publiczny zostanie skompromitowany, wówczas należy niezwłocznie zgłosić do CA chęć unieważnienia wystawionego certyfikatu (Revocation Request). Proces ten ma również miejsce w przypadku zmiany danych w certyfikacie (dane adresowe, zmiana nazwiska, zmiana nazwy firmy, itp.) Informacja o unieważnieniu musi być upubliczniona przez CA lub odpowiedniego wystawcę CRL. Unieważnienie jest publikowane poprzez listę CRL, częstotliwość publikacji list zależy od polityki centrum.
- **Certyfikacja skrośna**
  Centra certyfikacji mogą certyfikować wzajemnie swoje klucze publiczne (Cross-Certification). Certyfikacja skrośna moze być dwukierunkowa (centra certyfikują się wzajemnie) lub jednokierunkowa, wtedy nadrzędne centrum CA_1 certyfikuje centrum podrzędne CA_2, tworząc hierarchię centrów.

- **Dodatkowe funkcje**
  - informacja o zmianie klucza CA (CA key update announcement)
  - ogłoszenie certyfikatu (Certificate announcement) — ogłoszenie utworzenia certyfikatu
  - ogłoszenie unieważnienia (Revocation announcement) — poinformowanie użytkownika, że jego certyfikat został (lub zostanie niedługo) unieważniony
  - ogłoszenie listy CRL (CRL announcement) — ogłoszenie opublickowania nowej listy CRL
  - potwierdzenie certyfikatu (Certificate confirmation) — EE ogłasza przyjęcie lub odrzucenie wystawionego certyfikatu
  - archiwizacja kluczy (Key Archive) — jawne żądanie przechowywania przez centrum klucza prywatnego użytkownika

Centra certyfikacji w Polsce

- NBP Narodowe Centrum Certyfikacji (NCCert)
- Certum — Powszechne Centrum Certyfikacji
- PWPW Sigillum — Polskie Centrum Certyfikacji Elektronicznej

Zagrożenia

- Ataki komputerowe
  - podkładanie fałszywych Root CA w przeglądarkach
  - podkładanie fałszywych stron (phishing, pharming)
- atak urodzinowy na MD5
- niektóre implementacje X.509 nie sprawdzają zastosowania klucza (podpis, szyfrowanie)

Inne rozwiązania

Web of trust PGP (Pretty Good Privacy, GNU Privacy Guard)

- Każdy jest zaufaną trzecią stroną, otrzymuje key-ring
- Identyfikujemy osoby po adresie e-mail
- Zalety: niskie koszty, odporność na atak
- Wady: aktualizacja, problemy w dużych systemach