Sprzętowe wspomaganie zabezpieczania informacji

Sprzętowe wspomaganie zabezpieczania informacji

Podczas realizacji laboratorium posługujemy się wielofunkcyjnym urządzeniem zabezpieczającym YubiKey 5, obsługującym szereg funkcjonalności oraz standardów związanych ze sprzętowym zabezpieczeniem informacji.

Urządzenie posiada:

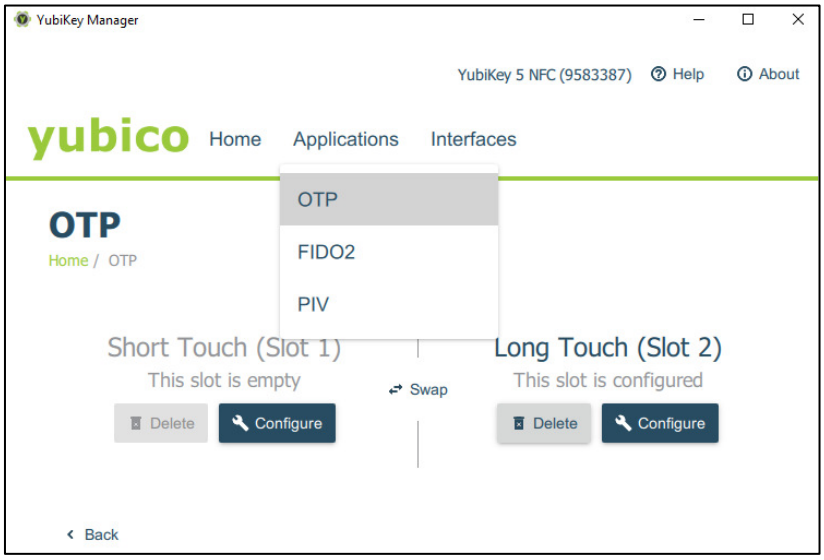
- Dwa sloty aplikacyjne umożliwiające uruchomienie aplikacji pozwalających na obsługę jednego z następujących rodzajów haseł:
  - Haseł statycznych (Static password).
  - Haseł jednorazowych, generowanych:
    - firmowym mechanizmem Yubico OTP,
    - zgodnie ze standardem OATH – haseł HOTP i (we współpracy z oprogramowaniem w systemie operacyjnym) TOTP.
  - Uwierzytelniania w trybie challenge – response:
    - Przy wykorzystaniu firmowego mechanizmu Yubico OTP,
    - z użyciem standardu skrótu HMAC-SHA1.
- Możliwość uwierzytelniania zgodnie ze standardami Fast IDentity Online (FIDO) Alliance:
  - Universal 2-Factor (U2F),
  - Fast IDentity Online 2 (FIDO2) – wstecznie kompatybilnym z U2F.
- Możliwość pracy jako karta inteligentna w standardzie PIV (Personal Identity Verification).
- Możliwość pracy jako karta OpenPGP.

Konfiguracja slotów aplikacyjnych i obsługa urządzenia YubiKey

Slot 1: Wywoływany krótkim przyciśnięciem przycisku na tokenie. Używany zwykle do generowania OTP. Domyślnie ustawiona opcja **append-cr**.

Slot 2: Wywoływany długim przytrzymaniem przycisku na tokenie. Używany zwykle do przenoszenia statycznych kluczy. Domyślnie ustawione opcje **append-cr**, **static-ticket**, **strong-pw1**, **strong-pw**2, **man-update**.

Do konfiguracji slotów używamy programu **YubiKey Manager (Applications->OTP)** lub polecenia **ykpersonalize**.



Po wybraniu opcji **Applications->OTP** program wyświetla informację czy dany slot jest już skonfigurowany (empty/configured) oraz daje możliwość:

- **Delete** – usunięcia aplikacji z danego slotu,
- **Configure** – skonfigurowania aplikacji określonego typu (Yubico OTP, Static Password, Challenge-response, OATH-HOTP) w danym slocie.

Niektóre opcje konfiguracyjne nie są dostępne z użyciem programu YubiKey Manager, lecz mogą być zmieniane z linii poleceń, przy wykorzystaniu polecenia **ykpersonalize**.

ykpersonalize

Polecenie **ykpersonalize** umożliwia zmiankę konfiguracji urządzenia YubiKey, w tym konfiguracji slotów aplikacyjnych.

```
ykpersonalize {-1 | -2} [-o<opcja>] -a <klucz tajny 16 lub 20 bajtów hex>
```

Parametr **-1** lub -**2** określa skot aplikacyjny który konfigurujemy.

W przypadku polecenia **ykpersonalize** w systemie Windows zawsze konieczne jest podanie klucza tajnego z użyciem parametru **-a** – jest możliwe wygenerowanie go w sposób automatyczny.

Nazwę opcji do włączenia podajemy po parametrze -o bez rozdzielającej spacji (np.: -ostatic-ticket). Większość z poniższych opcji może zostać poprzedzona znakiem **'-'** (minus, np. -o-append-cr), co oznacza jej wyłączenie. Może to być konieczne w celu wyłączenia opcji domyślnie włączonych dla danego slotu. Parametr -o można podać kilkukrotnie w jednym poleceniu.

Opcje określające aplikację wykorzystywaną w danym slocie:

- **static-ticket** – hasło statyczne,
- **oath-hotp** – hasła jednorazowe typu HOTP w standardzie OATH,
- **chal-resp** – generowanie odpowiedzi w trybie challenge-response.

Opcje dla aplikacji OATH-HOTP:

- **tab-first** – wyślij TAB na początku hasła,
- **append-tab1** – jeśli używane jest OTP zawierające część stałą i zmienną, wyślij TAB po stałej,
- **append-delay1** – odczekaj ½ s przed wysłaniem zmiennej części,
- **append-tab2** – wyślij TAB po wysłaniu hasła,
- **append-delay2** – odczekaj ½ s po wysłaniu hasła,
- **append-cr** – wyślij ENTER po wysłaniu hasła,
- **oath-hotp8** – generuj hasła długości 8 cyfr (domyślnie 6 cyfr).

Opcje dla aplikacji challenge-response:

- **chal-yubico** – sposób generowania odpowiedzi: wg firmowego rozwiązania YubiCo,
- **chal-hmac** – sposób generowania odpowiedzi: wg standardu HMAC-SHA1,
- **chal-btn-trig** – wymagaj potwierdzenia działania przyciskiem na tokenie.

ykchalresp

Polecenie **ykchalresp** pozwala na uzyskanie od urządzenia YubiKey odpowiedzi wykorzystywanej podczas uwierzytelniania w trybie challenge-response.

`ykchalresp {-1 | -2} [-H | -Y] [-x] [-6 | -8] <challenge>`

Parametr **-1** lub **-2** oznacza numer slotu aplikacyjnego który ma być wykorzystany przy generowaniu odpowiedzi na challenge.

Rodzaj odpowiedzi:

- **-H** (domyślnie) – 64 bajtowa odpowiedź wg standardu HMAC-SHA1,
- **-Y** – 6 bajtowa odpowiedź wg firmowego rozwiązania YubiCo, emulującego podejście OTP.
- **-x** – określa, że challenge przekazujemy w formacie heksadecymalnym,
- **-6** – wymusza format odpowiedzi zgodny z 6 cyfrowym formatem OATH,
- **-8** – wymusza format odpowiedzi zgodny z 6 cyfrowym formatem OATH.

Universal 2nd-factor Authentication (U2F)

U2F jest rozwiązaniem opracowanym przez FIDO Alliance, pozwalającym na wykorzystanie dedykowanego urządzenia w procesie uwierzytelniania w aplikacjach WWW. Rozwiązanie to ma stanowić dodatek/uzupełnienie innego mechanizmu weryfikacji tożsamości (multistep authentication) – np. uwierzytelniania z użyciem nazwy użytkownika i hasła. Do działania mechanizmu niezbędna jest wcześniejsza identyfikacja próbującego uzyskać dostęp użytkownika.

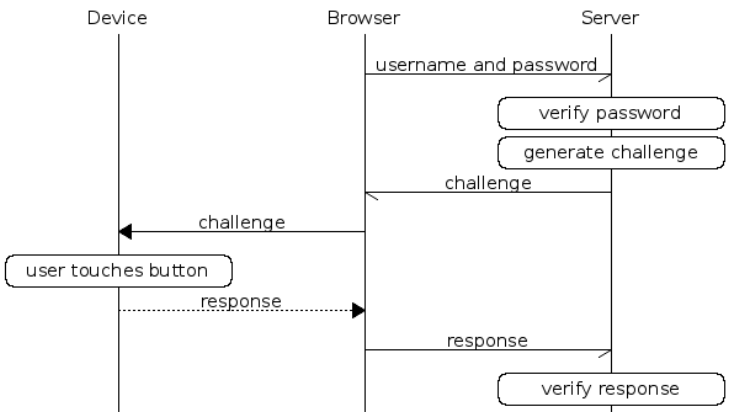
Specyfikacja mechanizmu wprowadza wyraźny podział na następujące elementy tego procesu:

- stronę zainteresowaną potwierdzeniem tożsamości użytkownika – np. serwer WWW (server, relaying party, RP),
- oprogramowanie po stronie użytkownika, służące mu do interakcji z aplikacją (browser, client),
- urządzenie U2F pozwalające na potwierdzenie tożsamości użytkownika (device, U2F device).

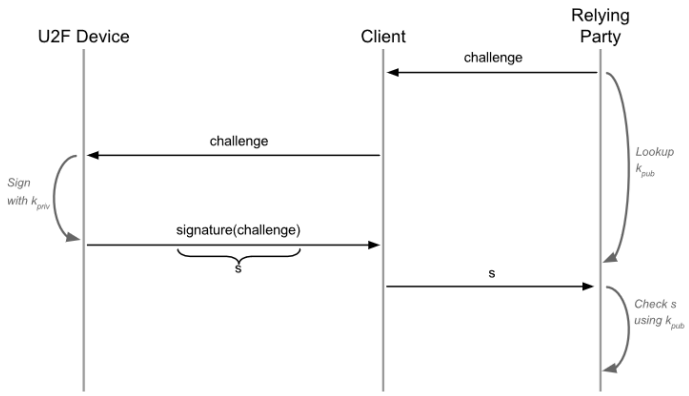
Komunikacja pomiędzy klientem (np. przeglądarką internetową), a urządzeniem uwierzytelniającym (urządzeniem U2F) odbywa się wg zasad określonych protokołem Client to Authenticatior Protocol (CTAP) standaryzowanym przez FIDO Alliance. Dopuszczalna jest komunikacja przez USB, NFC i Bluetooth. Rozwiązanie jest natywnie obsługiwane przez popularne przeglądarki internetowe – jako U2F lub dzięki obsłudze nowszego rozwiązania FIDO2, które jest wstecznie kompatybilne z U2F.

Ogólny przebieg procesu uwierzytelniania

U2F wykorzystuje mechanizm challenge-response oparty kryptografii asymetrycznej. Tożsamość zidentyfikowanego wcześniej (np. przez podanie jego nazwy lub pełne uwierzytelnienie z użyciem innego mechanizmu) użytkownika potwierdzana jest poprzez przesłanie mu porcji danych (challenge), które powinien cyfrowo podpisać z użyciem swojego klucza prywatnego (k_{priv}).



Przeglądarka użytkownika komunikuje się z urządzeniem U2F i przekazuje mu identyfikację użytkownika oraz otrzymany challenge. Urządzenie U2F wyszukuje klucz prywatny danego użytkownika, podpisuje otrzymany challenge i zwraca wynik (response) przeglądarce. Wykonanie podpisu może wymagać akcji ze strony użytkownika, takiej jak np. naciśnięcie przycisku na urządzeniu U2F w celu potwierdzenia że jest on świadomy realizacji procesu uwierzytelnienia (tzw. proof-of-presence). Klucz k_{priv} nie jest możliwy do odczytania z urządzenia U2F – można jedynie wykorzystać go do wykonania podpisu cyfrowego.



Otrzymawszy podpisaną odpowiedź (s) od urządzenia U2F, przeglądarka odsyła ją do RP. Ponieważ użytkownik jest już uwierzytelniony lub co najmniej zidentyfikowany przez RP, jest ono w stanie wybrać odpowiedni klucz publiczny (k_{pub}) i użyć go do weryfikacji poprawności otrzymanego podpisu – jeśli jest on prawidłowy, uwierzytelnienie U2F kończy się pomyślnie.

Należy zwrócić uwagę, iż zgodnie ze swoją nazwą U2F wymaga w procesie uwierzytelniania wcześniejszej identyfikacji lub uwierzytelnienia z użyciem innego mechanizmu.

Proces rejestracji

Wspomniane klucze publiczny i prywatny, generowane są przez urządzenie uwierzytelniające U2F w procesie rejestracji. Gdy uwierzytelniony już innym mechanizmem użytkownik danej aplikacji WWW zechce umożliwić potwierdzenie w niej swojej tożsamości przy użyciu danego urządzenia U2F, urządzenie to generuje parę kluczy publiczny-prywatny (k_{pub}-k_{priv}) i przekazuje klucz publiczny (k_{pub}) klientowi (np. przeglądarce internetowej), który z kolei dostarcza go aplikacji (RP).

RP zapamiętuje otrzymany klucz i jego powiązanie z tożsamością użytkownika (a właściwie z otrzymaną wartością parameteru key_handle – patrz rozdział następny). Dzięki temu, w przyszłości będzie w stanie poprawnie wybrać klucz publiczny potrzeby do weryfikacji podpisanej przez urządzenie U2F wiadomości uwierzytelniającej.

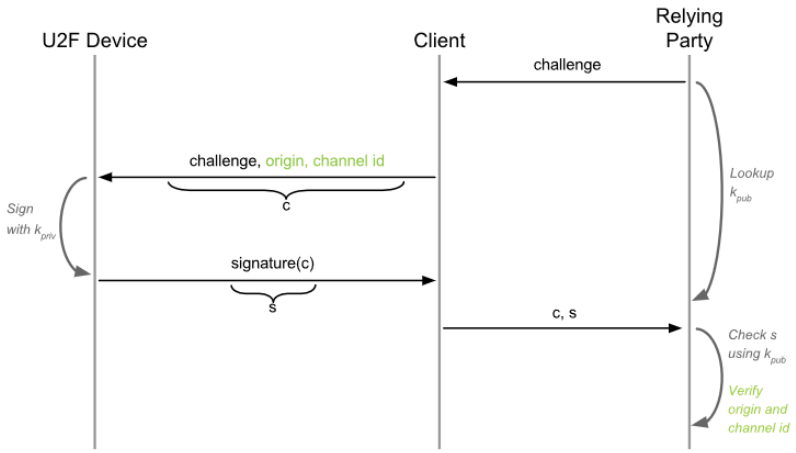
Klucz prywatny nigdy nie opuszcza urządzenia uwierzytelniającego i nie jest możliwy do odczytania – urządzenie pozwala jedynie podpisywanie wiadomości uwierzytelniających (challenge) z jego użyciem.

Powyższy sposób realizacji procesu rejestracji/uwierzytelniania powoduje, że klasyczne techniki ataku takie jak podsłuchiwanie wiadomości wymienianych w procesie uwierzytelniania czy poznania treści bazy danych uwierzytelniających po stronie serwera nie skutkują możliwością podszywania się pod użytkownika.

Dodatkowe zabezpieczenia i funkcjonalności w procesie uwierzytelniania U2F

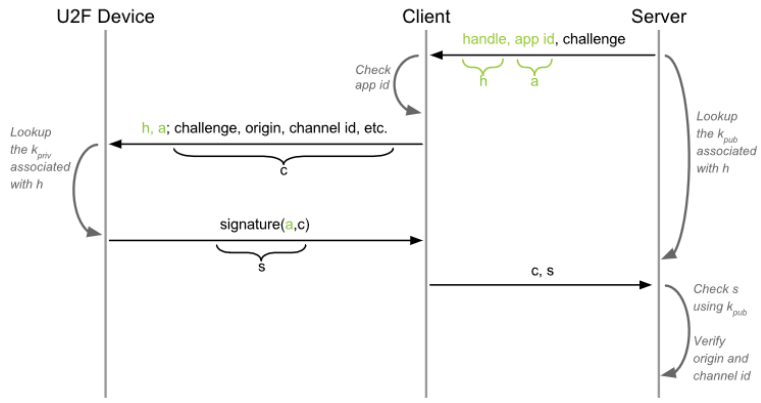
Powyższe rozwiązanie nie zapewniałoby ochrony przed atakami typu pishing i man-in-the-middle, zatem zakłada się, iż uwierzytelnianie realizowane przy wykorzystaniu połączenia protokołu TLS

nawiązanego pomiędzy RP, a przeglądarką po stronie klienta. Protokół TLS jest w stanie potwierdzić tożsamość RP wykorzystując jego certyfikat (ochrona przed atakami typu pishing). Dodatkowo TLS utrzymuje informacje pozwalające jednoznacznie zidentyfikować dane połączenie z określonym RP. Tożsamość RP oraz identyfikacja połączenia TLS są dołączane do żądania przesyłanego przez przeglądarkę do urządzenia U2F np. i stają się tym samym częścią podpisanej przez nie odpowiedzi (c,s).



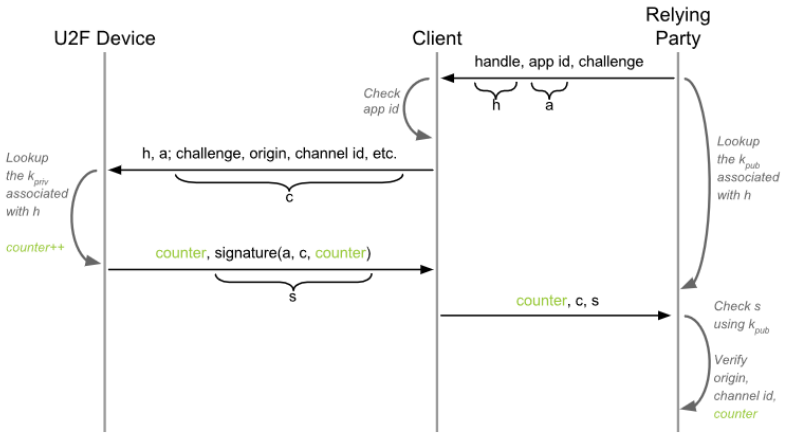
Ich obecność w odpowiedzi otrzymanej przez RP pozwala mu porównać ich wartości z identyfikacją połączenia które RP utrzymuje z klientem – jeśli będą takie same, to połączenie RP z klientem jest bezpośrednie; jeśli będą się różnić, oznacza to istnienie obiektu pośredniczącego w komunikacji pomiędzy RP i klientem (terminującego połączenie z RP i zestawiającego nowe do klienta).

Twórcy rozwiązania U2F przewidzieli możliwość istnienia wielu aplikacji na określonym serwerze, a także posiadanie przez jednego użytkownika (właściciela urządzenia U2F) wielu kont do tej samej aplikacji. Aby umożliwić współistnienie wielu aplikacji oraz uniemożliwić RP powiązanie ze sobą wielu kont tego samego użytkownika wprowadzono identyfikatory app_id (a) oraz key_handle (h).



Identyfikator key_handle generowany jest przez urządzenie U2F podczas rejestracji i przekazywany RP wraz z kluczem publicznym (k_{pub}). Podczas procesu uwierzytelnienia identyfikatory app_id (identyfikujący określoną aplikację, np.: jej URI) oraz key_handle (wartość binarna) są przekazywane do urządzenia U2F, umożliwiając mu odszukanie właściwego klucza prywatnego w celu wykonania podpisu. Dzięki temu, że właściwy klucz jest identyfikowany z użyciem wartości key_handle, generowanej losowo przy każdej rejestracji, nie sposób powiązać ze sobą różnych kont do których uwierzytelniamy się z użyciem tego samego urządzenia U2F.

Pomimo komplikacji takiego procesu, należy założyć, iż możliwe jest skopiowanie urządzenia U2F jeśli ktoś ma do niego fizyczny dostęp. Aby zabezpieczyć się przed wykorzystaniem kopii urządzenia U2F, utrzymuje ono wewnętrzny licznik przeprowadzonych prób uwierzytelnienia, którego wartość dodawana jest do przesyłanej do RP, podpisanej wiadomości uwierzytelniającej.

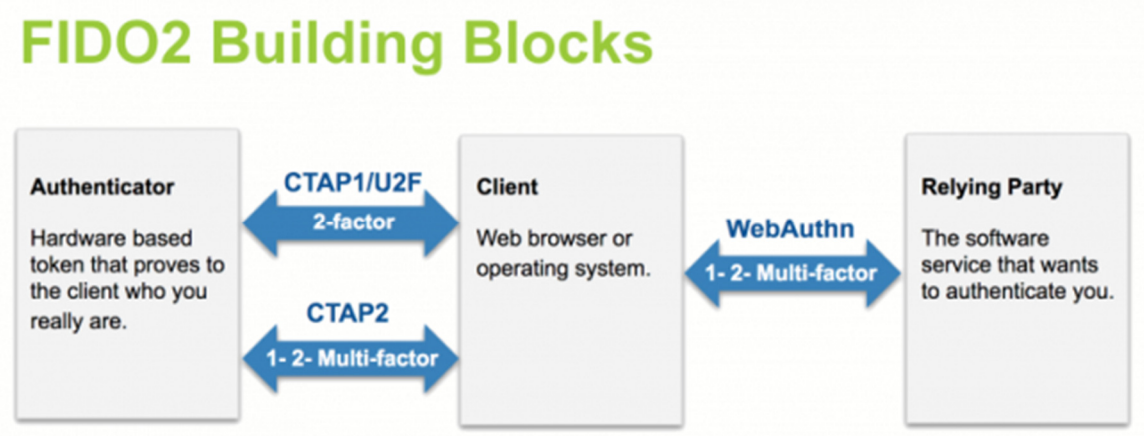


Jeśli RP wykryje, że odpowiedź urządzenia zawiera wartość licznika która już wcześniej wystąpiła, oznacza to istnienie kopii danego urządzenia. Standard U2F nie precyzuje, jakie działania należy podjąć w danym wypadku.

Fast IDentity Online 2 (FIDO2)

Rozwiązanie FIDO2 powstało w wyniku rozwoju rozwiązania U2F i jest z nim wstecznie kompatybilne. Łączy ono 2 mechanizmy:

- Client to Authenticatior Protocol 2 (CTAP2) – protokół do komunikacji pomiędzy klientem (np. przeglądarką), a urządzeniem uwierzytelniającym,
- WebAuthn – API standaryzowane przez W3C i odpowiedzialne za komunikację oraz przeprowadzenie procesu uwierzytelniania pomiędzy klientem a RP.



Ogólny sposób funkcjonowania rozwiązania FIDO2 zbliżony jest do opisanego wcześniej U2F, jednakże oferowana jest ponadto możliwość przeprowadzenia uwierzytelniania użytkownika z użyciem wyłącznie mechanizmów FIDO2, bez uprzedniej identyfikacji (lub uwierzytelnienia) innym mechanizmem. Pozwala to w łatwy sposób wyeliminować hasła z procesu uwierzytelniania do aplikacji

WWW. FIDO2 jest obecnie obsługiwane natywnie przez popularne przeglądarki internetowe, takie jak Chrome, Firefox i Edge.

Ograniczenia w pojemności urządzeń U2F/FIDO2

Ponieważ zarówno U2F jak i FIDO2 zakładają wygenerowanie unikalnej pary kluczy asymetrycznych dla każdej tożsamości wykorzystywanej przez użytkownika w każdej z aplikacji WWW z osobna (jedna tożsamość nie może być współdzielona przez wiele aplikacji), liczba kluczy prywatnych które użytkownik chciałby przechowywać w urządzeniu uwierzytelniającym może być znaczna. Niestety, ograniczenia pamięciowe popularnych urządzeń uwierzytelniających powodują, że możliwe jest przechowywanie stosunkowo małej ich liczby – dla urządzenia YubiKey 5 jest to 25.

Z tego względu firma YubiCo zdecydowała się na zaimplementowanie rozwiązania pośredniego, które umożliwia wprawdzie wykorzystanie urządzeń YubiKey w powiązaniu z nieograniczoną liczbą tożsamości użytkownika, lecz dzieje się to kosztem rezygnacji z szeregu opisanych wyżej zalet rozwiązań U2F i FIDO2.

Rozwiązanie to działa analogicznie do opisanego powyżej rozwiązania U2F/FIDO2, lecz wygenerowany podczas rejestracji klucz prywatny (k_{priv}) jest szyfrowany przy wykorzystaniu kodera AES-256 w trybie CCM i przesyłany do RP jako identyfikator key_handle (U2F) lub ID (FIDO2) wraz z kluczem publicznym (k_{pub}). Symetryczny klucz AES użyty do zaszyfrowania klucza k_{priv} znany jest wyłącznie urządzeniu uwierzytelniającemu YubiKey. Jest on generowany losowo w procesie pierwszego uruchomienia urządzenia lub przy resecie ustawień U2F/FIDO2 do ustawień fabrycznych.

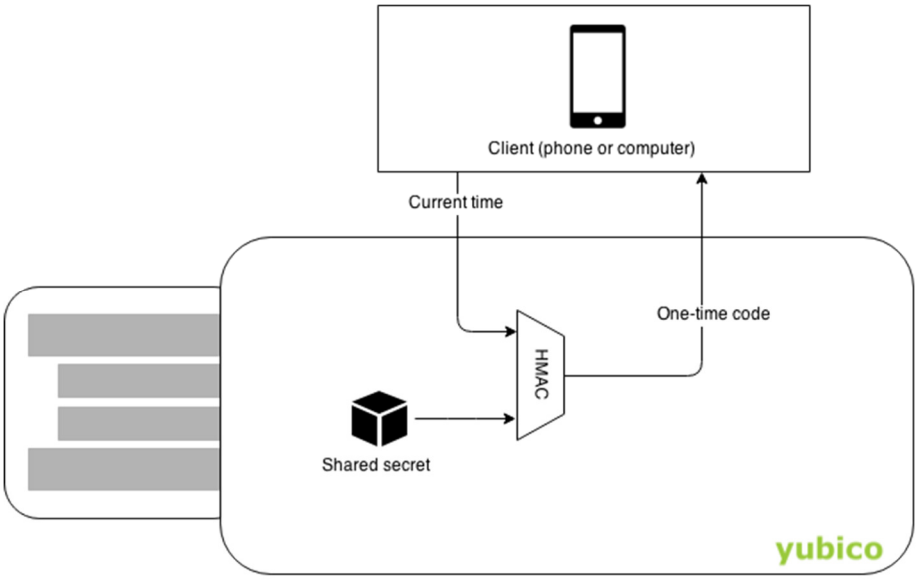
W ten sposób RP uzyskuje klucz k_{pub} oraz związaną z nim wartość key_handle/id będącą w istocie zaszyfrowanym kluczem k_{priv}. W procesie uwierzytelnienia RP przesyła do klienta, a ten do urządzenia uwierzytelniającego dane challenge oraz wartość key_handle/id, która z założenia ma umożliwić urządzeniu uwierzytelniającemu odszukanie w swojej pamięci odpowiedniego k_{priv}. W tym jednak wypadku urządzenie YubiKey odszyfrowuje po prostu wartość otrzymaną jako key_handle/id, uzyskując k_{priv}, wykorzystywany następnie do podpisania odpowiedzi na challenge.

Mechanizm ten powoduje, że urządzenie uwierzytelniające nie musi przechowywać żadnych kluczy prywatnych, gdyż są mu one dostarczane przez RP. Jednocześnie fakt, iż jedynie dane urządzenie uwierzytelniające zna symetryczny klucz pozwalający na prawidłowe odszyfrowanie otrzymanych w key_handle/id kluczy prywatnych, ma gwarantować bezpieczeństwo procesu uwierzytelnienia. Należy jednak zauważyć, że rozwiązanie to powoduje, iż baza danych uwierzytelniających po stronie RP ponownie staje się potencjalnie atrakcyjnym obiektem ataku.

Hasła jednorazowe OATH-TOPT

W przypadku urządzeń uwierzytelniających pozbawionych stałego zasilania, takich jak np. wykorzystywanego podczas laboratorium urządzenia YubiKey, nie jest możliwa realizacja funkcjonalności generatora zmiennych czasowo haseł jednorazowych OATH-TOPT (Time OTP) gdyż nie posiadają one zegara mogącego kontrolować ten proces.

Ograniczenie to można pokonać dzięki współpracy urządzenia uwierzytelniającego posiadającego możliwość generowania haseł jednorazowych których zmiana odbywa się przy próbie uwierzytelnienia (Event-based OATH-HOTP) z aplikacją uruchomioną na urządzeniu posiadającym zegar czasu rzeczywistego (np. komputerze, tablecie, telefonie).

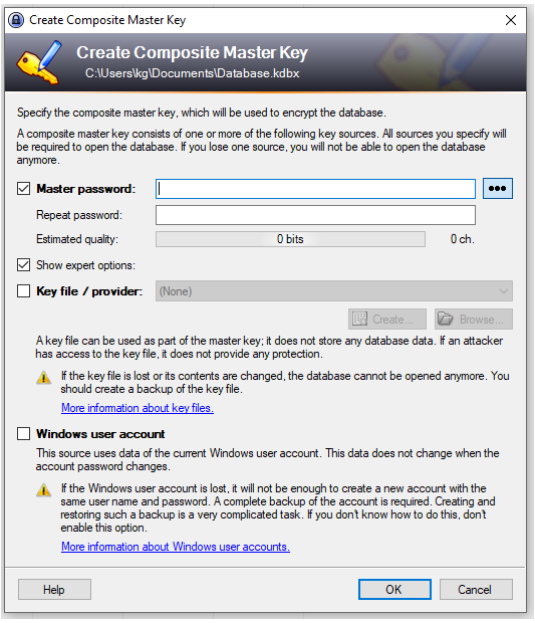


Oprogramowanie KeePass 2

Program KeePass 2 pozwala na przechowywanie informacji poufnych w zabezpieczonym kryptograficznie pliku (np. zaszyfrowanym jednym/kilkoma z dostępnych algorytmów). Dostęp do powyższych informacji możliwy jest po przeprowadzeniu uwierzytelnienia jedną z dostępnych metod. Metody te obejmują np.:

- hasło (master password – domyślnie),
- plik z określoną, statyczną zawartością (keyfile),
- uwierzytelnienie jako określony użytkownik systemu operacyjnego (Windows user account),
- inne metody zaimplementowane w postaci wtyczek (plugins/providers).

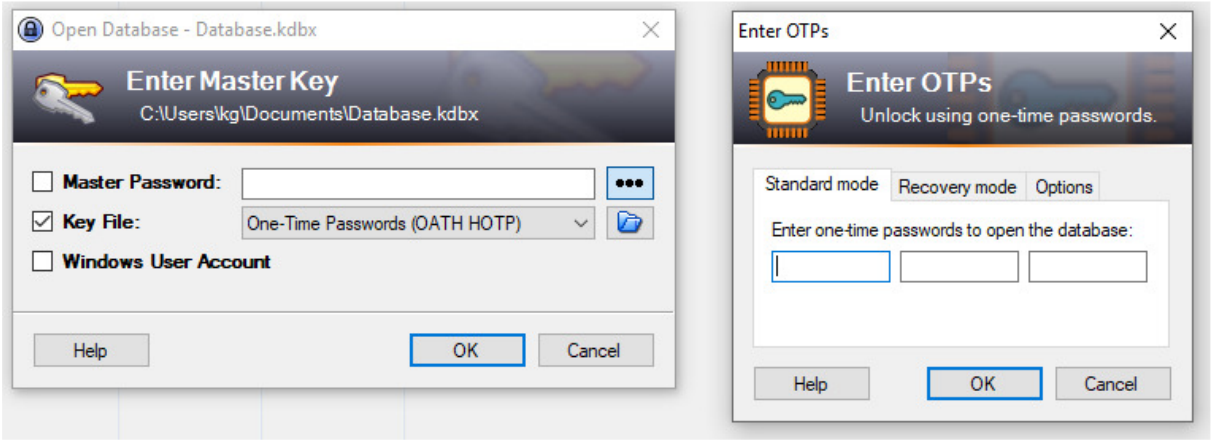
Metodę potwierdzenia tożsamości użytkownika niezbędną do otwarcia określonej bazy danych wybieramy przy jej tworzeniu (File->New).



Interesujące nas dodatkowe metody uwierzytelniania (providers) dostępne w postaci wtyczek, możliwych do wybrania w sekcji **Key file / Provider**, dostępnej po zaznaczeniu opcji **Show expert options**.

Wybranie tego rodzaju metody spowoduje pojawienie się dodatkowego okna dialogowego po zatwierdzeniu ustawień w obecnym oknie. Dodatkowe okno dialogowe umożliwia konfigurację parametrów działania wybranej metody.

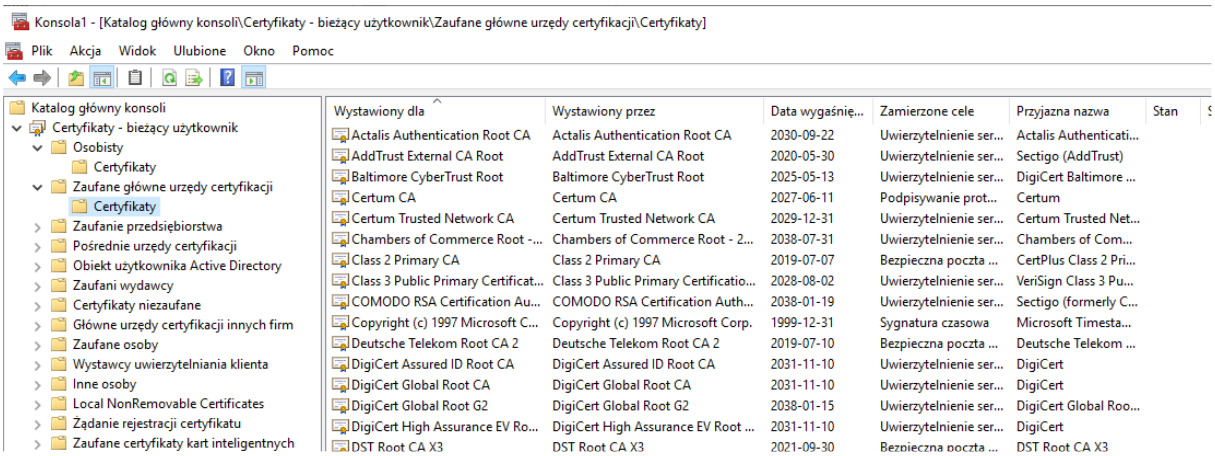
Przy próbie otwarcia pliku zabezpieczonej bazy danych pojawi się okno dialogowe umożliwiające podanie wymaganych informacji uwierzytelniających, różnych w zależności od sposobu je zabezpieczenia.



Dostęp do magazynu certyfikatów systemu Windows

Dostęp do magazynu certyfikatów w systemie Windows można uzyskać używając polecenia **mmc.exe**. Spowoduje ono pojawienie się pustej konsoli zarządzania, do której dodajemy (**Plik->Dodaj/usuń przystawkę**) przystawkę **Certyfikaty (Certificates)**. Próba dodania powyższej przystawki spowoduje pojawienie się okna z pytaniem, czyje certyfikaty mają być z jej użyciem zarządzane – wybieramy **Moje konto użytkownika**.

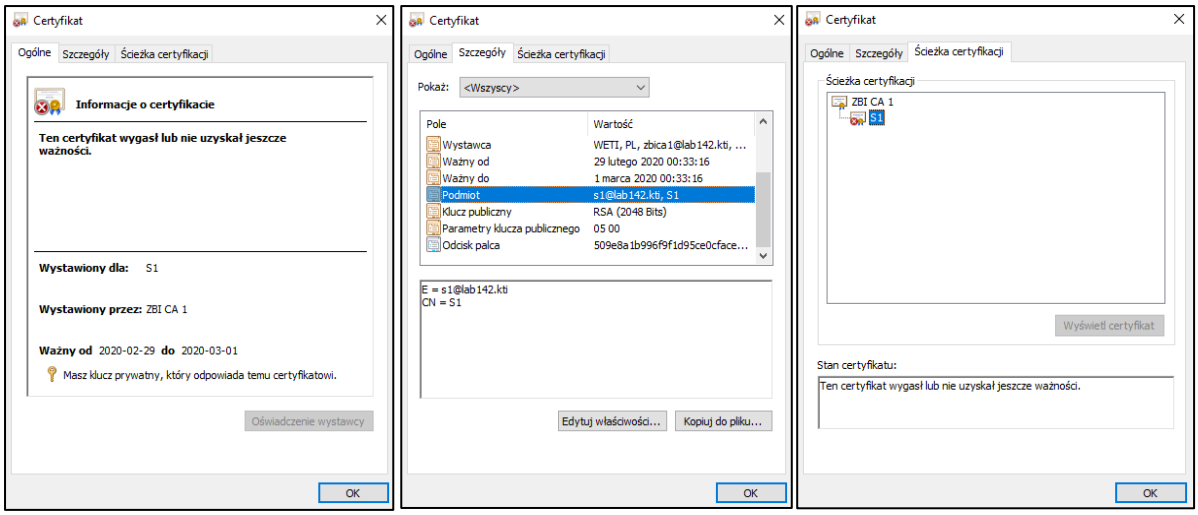
Po dodaniu powyższej przystawki do konsoli zarządzania, umożliwia ona przeglądanie certyfikatów danego użytkownika, podzielonych na kategorie (lista po lewej).



Na szczególną uwagę zasługują kategorie:

- **Osobisty** – certyfikaty, których właścicielem jest dany użytkownik,
- **Zaufane główne urzędy certyfikacji** – urzędy certyfikacji, którym dany użytkownik ufa (korzenie drzew zaufania).

Powyższa konsola pozwala na wiele operacji na zbiorze certyfikatów (menu Akcje lub menu kontekstowe), np.: usuwanie, importowanie, tworzenie żądań itp.



Możliwe jest też przeglądanie szczegółów certyfikatu – można tu np. znaleźć informację czy dany użytkownik posiada klucz prywatny odpowiadający danemu certyfikatowi, jakie są rozszerzenia certyfikatu i jak wygląda ścieżka certyfikacji. Podana też będzie informacja czy dany certyfikat został uznany za ważny, a jeśli nie, to dlaczego.

Aby dany certyfikat mógł zostać wykorzystany do ochrony wiadomości email zgodnie ze specyfikacją S/MIME, musi on (najczęściej – zależne od aplikacji) spełniać następujące warunki:

- certyfikat jest uznany za prawidłowy przez system operacyjny,
- użytkownik posiada klucz prywatny odpowiadający certyfikatowi – do podpisywania i odszyfrowania,
- emailAddress zgodny z adresem nadawcy,
- prawidłowe keyUsage (najczęściej: digitalSignature, nonRepudiation, keyEncipherment),
- prawidłowe extendedKeyUsage (emailProtection).

W przypadku wykorzystania kart inteligentnych pracujących w standardzie PIV, zawarte na nich certyfikaty są dodawane (kopiowane) do magazynu systemu Windows w momencie ich podłączenia/włożenia do czytnika. Klucze prywatne pozostają na karcie. Oznacza to, że po zaimportowaniu certyfikatu do klucza sprzętowego/karty, należy je wyjąć i ponownie podłączyć/włożyć do czytnika, zanim certyfikaty te staną się widoczne.

Informacje dotyczące obsługi standardu PIV w urządzeniu YubiKey 5:

https://developers.yubico.com/PIV/