Работа с политиками PIN-кодов в PKCS#11

В Рутокен ЭЦП 3.0 есть настраиваемые аппаратные политики качества PIN-кодов, которые обрабатываются микропрограммой при соответствующих операциях.

Чтение и задание политик качества PIN-кодов при смене

Для работы с политиками PIN-кодов был добавлен объект политик PIN-кодов типа CKO_HW_FEATURE, который можно найти с помощью функций `C_FindObject*` по шаблону с помощью специальных атрибутов:

Шаблон получения объекта политик PIN-кодов

```
CK_OBJECT_CLASS hwFeatureClass = CKO_HW_FEATURE;
CK_HW_FEATURE_TYPE pinPolicyFeatureType = CKH_VENDOR_PIN_POLICY;
CK_USER_TYPE userType = CKU_USER;

CK_ATTRIBUTE[] pinPolicyTemplate {
    { CKA_CLASS, &hwFeatureClass, sizeof(hwFeatureClass) },
    { CKA_HW_FEATURE_TYPE, &pinPolicyFeatureType, sizeof(pinPolicyFeatureType) },
    { CKA_VENDOR_USER_TYPE, &userType, sizeof(userType) },
};
```

Для каждого подключенного Рутокена (даже тех, которые не поддерживают политики PIN-кодов) найдется ровно один объект. Для получения данного объекта – быть авторизованным на Рутокене – не обязательно.

Чтобы узнать поддерживает ли токен политики PIN-кодов – у объекта политик PIN-кодов надо запросить атрибут `CKA_VENDOR_SUPPORTED_PIN_POLICIES` с помощью функции C_GetAttributeValue. Если данный массив будет пустым (иметь нулевую длину), значит политики PIN-кодов не поддерживаются токеном.

Чтение и задание политик **качества PIN-кодов смены PIN-кодов** реализована через функции `C_GetAttributeValue` и `C_SetAttributeValue` о бъекта политик PIN-кодов.
Для задания политик нужно предъявить PIN-код Администратора в `C_Login`.
Читать политики возможно без предъявления прав Пользователя или Администратора.

Если при вызове `C_SetPIN` устанавливаемый PIN-код не удовлетворяет установлены политикам для выбранного пользователя – вернется ошибка `CKR_INAPPROPRIATE_PIN` или `CKR_PIN_IN_HISTORY`.

Варианты содержимого шаблона:

Название	Тип	Обязательный атрибут	Значение	Атрибут только для чтения
`CKA_CLASS`	`CK_OBJECT_CLASS`	+	`CKO_HW_FEATURE`	+
`CKA_HW_FEATURE_TYPE`	`CK_HW_FEATURE_TYPE`	+	`CKH_VENDOR_PIN_POLICY`	+
`CKA_VENDOR_USER_TYPE`	`CK_USER_TYPE`	+	Пользователь, которому соответствует объект политик	+
`CKA_VENDOR_PIN_POLICY_STATE`	`CK_ULONG`	-	• `PIN_POLICY_STATE_WELL_DEFINED` - все атрибуты, которые позволяет задавать устройство, известны библиотеке. • `PIN_POLICY_STATE_HAS_UNKNOWN_ATTRIBUTES` - не все атрибуты, которые позволяет задавать устройство, известны библиотеке, но все они содержат значения по умолчанию. • `PIN_POLICY_STATE_HAS_UNKNOWN_NONDEFAULT_ATTRIBUTES` - не все неизвестные библиотеке атрибуты имеют значения по умолчанию.	+
`CKA_MODIFIABLE`	`CK_BBOOL`	-	`CK_TRUE`, если Администратор может изменять политики, иначе `CK_FALSE`. Возможна смена с `CK_TRUE` на `CK_FALSE`.	-

CKA_VENDOR_PIN_POLICIES_DELETABLE	CK_BBOOL	-	CK_TRUE, если политики будут удалены при форматировании, иначе CK_FALSE.	-
CKA_VENDOR_SUPPORTED_PIN_POLICIES	ARRAY (CK_ULONG)	-	Массив типов атрибутов, которые возможно выставить для данного типа пользователя на данном устройстве, используя данную версию библиотеки	+
CKA_VENDOR_PIN_POLICY_MIN_LENGTH	CK_BYTE	-	число, задающее минимальную длину PIN-кода	-
CKA_VENDOR_PIN_POLICY_HISTORY_DEPTH	CK_BYTE	-	число, задающее количество PIN-кодов, которое требуется хранить для невозможности смены на уже использованное значение	-
CKA_VENDOR_PIN_POLICY_ALLOW_DEFAULT_PIN_USAGE	CK_BBOOL	-	CK_TRUE, если разрешена смена PIN-кода на значение по умолчанию, иначе CK_FALSE	-
CKA_VENDOR_PIN_POLICY_DIGIT_REQUIRED	CK_BBOOL	-	CK_TRUE, если в PIN-коде требуется хотя бы одна цифра, иначе CK_FALSE	-
CKA_VENDOR_PIN_POLICY_UPPERCASE_REQUIRED	CK_BBOOL	-	CK_TRUE, если в PIN-коде требуется хотя бы один символ в заглавном регистре, иначе CK_FALSE	-
CKA_VENDOR_PIN_POLICY_LOWERCASE_REQUIRED	CK_BBOOL	-	CK_TRUE, если в PIN-коде требуется хотя бы один символ в строчном регистре, иначе CK_FALSE	-
CKA_VENDOR_PIN_POLICY_SPEC_CHAR_REQUIRED	CK_BBOOL	-	CK_TRUE, если в PIN-коде требуется хотя бы один специальный символ, иначе CK_FALSE	-
CKA_VENDOR_PIN_POLICY_DIFF_CHARS_REQUIRED	CK_BBOOL	-	CK_TRUE, если в PIN-коде требуется использование различных символов, иначе CK_FALSE	-

Пример работы с политиками PIN-кодов

Снизу приведен пример функции осуществляющий проверку поддержки политик PIN-кодов и получения атрибутов объекта политик PIN-кодов:

Работа с политиками PIN-кодов

```
int printPinPolicy(CK_SESSION_HANDLE session)
{
    CK_RV rv;
    CK_OBJECT_CLASS hwFeatureClass = CKO_HW_FEATURE;
    CK_HW_FEATURE_TYPE pinPolicyFeatureType = CKH_VENDOR_PIN_POLICY;
    CK_USER_TYPE userType = CKU_USER;

        //       PIN-
    CK_ATTRIBUTE getPinPolicyTemplate[] = {
        { CKA_CLASS, &hwFeatureClass, sizeof(hwFeatureClass) },
        { CKA_HW_FEATURE_TYPE, &pinPolicyFeatureType, sizeof(pinPolicyFeatureType) },
        { CKA_VENDOR_USER_TYPE, &userType, sizeof(userType) },
    };

        //    PIN-
    rv = fl->C_FindObjectsInit(session, getPinPolicyTemplate, sizeof (getPinPolicyTemplate) / sizeof
(getPinPolicyTemplate[0]));
    if (CKR_OK != rv) return rv;

    CK_OBJECT_HANDLE pinPolicyObject;
    CK_ULONG pinPolicyObjectCount;
    rv = fl->C_FindObjects(session, &pinPolicyObject, 1, &pinPolicyObjectCount);
    if (CKR_OK != rv) return rv;
    if (pinPolicyObjectCount != 1) return rv;

    rv = fl->C_FindObjectsFinal(session);
    if (CKR_OK != rv) return rv;

        //       PIN-
    CK_ATTRIBUTE supportPinPolicyTemplate[] = {
        { CKA_VENDOR_SUPPORTED_PIN_POLICIES, NULL_PTR, 0}
    };

        //    CKA_VENDOR_SUPPORTED_PIN_POLICIES
    rv = fl->C_GetAttributeValue(session, pinPolicyObject, supportPinPolicyTemplate, sizeof
```

```
(supportPinPolicyTemplate)/ sizeof(supportPinPolicyTemplate[0]));
    if (rv != CKR_OK) return rv;
        //  ,      PIN-
    if (supportPinPolicyTemplate[0].ulValueLen == 0) return -1;

    CK_BYTE minPinLength;
    CK_BYTE historyDepth;
    CK_BYTE allowDefaultPinUsage;
    CK_BYTE digitalRequired;
    CK_BYTE uppercaseRequired;
    CK_BYTE lowercaseRequired;
    CK_BYTE specCharsRequired;
    CK_BYTE diffCharsRequired;
    CK_BBOOL pinPolicyModifiable;
    CK_BBOOL pinPolicyDeletable;

        //        PIN-
    CK_ATTRIBUTE pinPolicyAttrs[] = {
        { CKA_VENDOR_PIN_POLICY_MIN_LENGTH, &minPinLength, sizeof(minPinLength)},
        { CKA_VENDOR_PIN_POLICY_HISTORY_DEPTH, &historyDepth, sizeof(historyDepth)},
        { CKA_VENDOR_PIN_POLICY_ALLOW_DEFAULT_PIN_USAGE, &allowDefaultPinUsage, sizeof(allowDefaultPinUsage)},
        { CKA_VENDOR_PIN_POLICY_DIGIT_REQUIRED, &digitalRequired, sizeof(digitalRequired)},
        { CKA_VENDOR_PIN_POLICY_UPPERCASE_REQUIRED, &uppercaseRequired, sizeof(uppercaseRequired)},
        { CKA_VENDOR_PIN_POLICY_LOWERCASE_REQUIRED, &lowercaseRequired, sizeof(lowercaseRequired)},
        { CKA_VENDOR_PIN_POLICY_SPEC_CHAR_REQUIRED, &specCharsRequired, sizeof(specCharsRequired)},
        { CKA_VENDOR_PIN_POLICY_DIFF_CHARS_REQUIRED, &diffCharsRequired, sizeof(diffCharsRequired)},
        { CKA_MODIFIABLE, &pinPolicyModifiable, sizeof(pinPolicyModifiable)},
        { CKA_VENDOR_PIN_POLICIES_DELETABLE, &pinPolicyDeletable, sizeof(pinPolicyDeletable)}
    };

        //      PIN-
    rv = fl->C_GetAttributeValue(session, pinPolicyObject, pinPolicyAttrs, sizeof(pinPolicyAttrs)/ sizeof
(pinPolicyAttrs[0]));
    if (CKR_OK != rv) return rv;

    printf("Min pin Length: %u\n", (unsigned) minPinLength);
    printf("History depth: %u\n", (unsigned) historyDepth);
    printf("Allow default Pin-code usage: %s\n", allowDefaultPinUsage == CK_TRUE? "true" : "false");
    printf("Pin-code requeres digits: %s\n", digitalRequired == CK_TRUE? "true" : "false");
    printf("Pin-code requeres uppercase chars: %s\n", uppercaseRequired == CK_TRUE? "true" : "false");
    printf("Pin-code requeres lowercase chars: %s\n", lowercaseRequired == CK_TRUE? "true" : "false");
    printf("Pin-code requeres spec chars: %s\n", specCharsRequired == CK_TRUE? "true" : "false");
    printf("Pin-code requeres different char usage: %s\n", diffCharsRequired == CK_TRUE? "true" : "false");
    printf("PIN-policy is modifiable by Admin: %s\n", pinPolicyModifiable == CK_TRUE? "true" : "false");
    printf("PIN-policy will be deleted after formating: %s\n", pinPolicyDeletable == CK_TRUE? "true" : "false");

    return rv;
}
```

Установка политик PIN-кодов выглядит схожим образом, за исключением того, что используется функция `C_SetAttributeValue` и переменные атрибутов инициализируются нужными значениями.

Функциональность принудительной смены PIN-кода

Администратор может запретить к использованию любой текущий PIN-код пользователя.
В `C_EX_TokenManage` есть режим: `MODE_FORCE_USER_TO_CHANGE_PIN`, тип `CK_USER_TYPE`.

Для выполнения команды требуется права Администратора.

Если выставлена принудительная смена, `C_Login` соответствующим пользователем возможен, но не доступны операции, которые требуют соответствующего пользователя.
Например, `C_Sign` и `C_EX_PKCS7Sign` будут возвращать ошибку `CKR_PIN_EXPIRED`.

Информацию о требовании смены PIN-кода допустимо получать в месте вызова пользователем `C_Login`.

Получение информации о необходимости смены PIN-кода

C_EX_SlotManage

Узнать о необходимости смены PIN-кода, можно с помощью функции C_EX_SlotManage в режиме MODE_GET_PIN_SET_TO_BE_CHANGED. Парам етром функции является CK_ULONG_PTR на идентификатор пользователя, для которого надо проверить требование смены ПИН-кода. Функция возвращает CKR_OK, если ПИН-код не нуждается в смене, и CKR_PIN_EXPIRED - иначе.