# COMARCH tPro ECC

## USER'S GUIDE

# TABLE OF CONTENTS

# LIST OF FIGURES

## Related documentation

| | |
|---|---|
| [RFC2616] | Fielding, R., Gettys, J., Mogul, J., Frystyk, H., *“Hypertext Transfer Protocol – HTTP/1.1”* RFC 2616, June 1999. |
| [RFC2437] | Kaliski, B., Staddon, J., *“PKCS#1: RSA Cryptography Specifications Version 2.0”* RFC 2437, October 1998. |
| [RFC3986] | Berners-Lee, T., Fielding, R., Masinter, L., *"Uniform Resource Identifier (URI): Generic Syntax"* RFC 3986, January 2005. |
| [RFC4492] | Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., Moeller, B., *“Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Security Layer (TLS)”* RFC 4492, May 2006. |
| [RFC5280] | Cooper, D., Santesson, S., Farell, S., Boeyen, S., Housley, R., Polk, W., *"Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile"* RFC 5280, May 2008. |
| [RFC6090] | McGrew, D., Igoe, K., Salter, M., *"Fundamental Elliptic Curve Cryptography Algorithms"* RFC 6090, February 2011. |
| [RFC6455] | Fette, I., Melnikov, A., *"The Websocket Protocol"* RFC 6455, December 2011. |
| [RFC7159] | Bray, T., *"The Javascript Object Notation (JSON) Data Interchange Format"* RFC 7159, March 2014. |
| [ISO9564-1] | *"Financial services – Personal Identification Number (PIN) management and security – Part 1: Basic principles and requirements for PINs in card-based systems"* ISO9564-1:2011, 2011. |

## Acronyms and abbreviations

| | |
|---|---|
| ECC | Elliptic Curve Cryptography, |
| RSA | Public-Key Cryptography Algorithm, |
| API | Application Programming Interface, |
| HTTP | Hypertext Transfer Protocol, |
| PIN | Personal Identification Number, |
| PUK | Personal Unblocking Key, |
| JSON | Javascript Object Notation, |
| POP | Proof Of Possession, |
| URL | Uniform Resource Locator, |
| PKI | Public Key Infrastructure. |

## Providing feedback about this document

Should you have any feedback considering this document, please contact our developers at tpro@comarch.com.

# Preface: About this document

## Introduction

Desired security levels in digital systems might be achieved in many ways. tPro Ecc is an easy-to-use token, which offers alternative approach for user authentication. It can be used both to authenticate user's identity while logging into various systems as well as in electronic banking and during online transactions execution. Following properties ensure comfort while using this token and complex solutions it provides:

- compatibility with smartphones, tablets, and PCs with operating systems such as: Windows, Linux and Mac,
- plug and play, driverless device,
- easy integration with other solutions,
- private keys never leave the device.

Complex functionalities, granted by tPro Ecc, are based on elliptic curves methods in asymmetric cryptography. Since the ECC offers security level comparable to the most popular RSA method, but with a much shorter keys, it is a very attractive algorithm when high performance asymmetric encryption is required (RSA algorithm is relatively slow) or when there is a very limited computing environment (such as smart cards).

Asymmetric cryptography is a way of encrypting information with the use of two keys: a public and a private one. The public key is accessible for anyone and is used to encrypt messages. The private key is secret, assigned to a particular person and is used to decrypt messages. Capturing of the public key will not enable the message decryption. It is also impossible to recreate the private key on the basis of the public key.

## Who should read it?

Content of this document is designed to be accessible by any person, who wishes to use tPro Ecc device. However, to access the whole spectrum of its usability, we encourage user to familiarize with knowledge concerning public-key cryptography and JS script language.

# 1. What functionalities does tPro Ecc provide?

For efficient usage of tPro Ecc token, there is no need to install any drivers, the device simply works without them. Its usability ranges for being able to operate with computer, tablet and smartphone. The device is designed in a way that makes it impossible for the private key, which is used to generate the passwords, to leave the token. Thanks to that, no one can copy the private key, even with access to the device's memory.

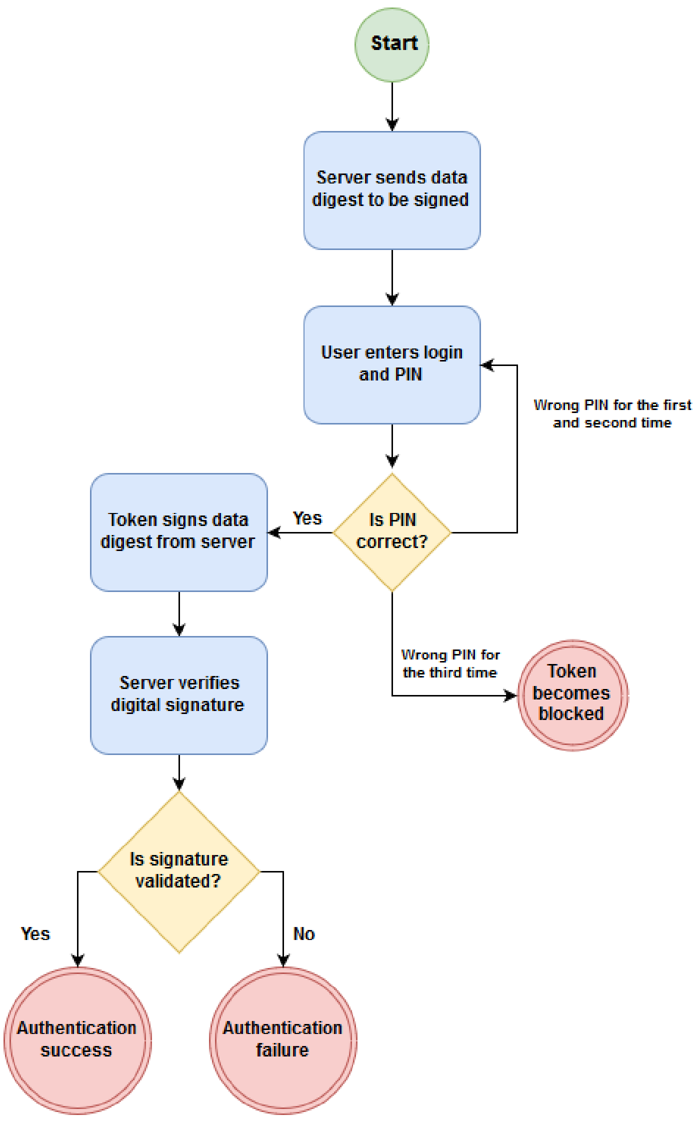
Start
Server sends data digest to be signed
User enters login and PIN
Wrong PIN for the first and second time
Is PIN correct?
Yes
Token signs data digest from server
Wrong PIN for the third time
Token becomes blocked
Server verifies digital signature
Is signature validated?
Yes
No
Authentication success
Authentication failure

*Figure 1. tPro Ecc default workflow.*

## 1.1. Digital signature of banking operations

Primal, although not the only one functionality provided by tPro Ecc token. Security in online transfers is a crucial issue for every institution, which offers this kind of service. Our solution provides mechanism of digital signature, which ensures data integrity during online transfers and non-repundation (entity authenticity). *Figure 1.* shows exact steps how to cope with this issue while using tPro Ecc token. With one little addition, data send from server to be signed ought to be based on each unique transaction information.

## 1.2. User authentication

The tPro Ecc token can be used in online banking, but not only. Besides transfers and orders approval, it can provide authentication while logging into enterprise systems. Generally, it has been designed to be applied wherever casual login and password authentication  is not enough.

Flow of action, in this case scenario, is very similar to digital signature of banking operations. Doing so might be performed by adjusting to the following steps:

a) Server generates random number,

b) User enters login,

c) User enters PIN assigned to owned tPro Ecc token,

d) tPro Ecc creates digital signature of provided random number and sends it back to the server,

e) Server verifies random number's signature using public key assigned to a specific user.

## 1.3. Storage of PKI Certificates

Additional functionality of tPro Ecc device is to store or return stored PKI certificates. For each of ECC key pairs, one certificate is made. Whenever application asks for token authentication, it might present specific certificate to do so.

# 2. How to perform tPro Ecc related actions?

Two types of API are provided for the developer to facilitate integration process. First API (client side) is for the browser usage – it allows for interaction between browser Javascript code and token itself. Second API (server side) is intended to be used at server side of the application, which is to be protected.

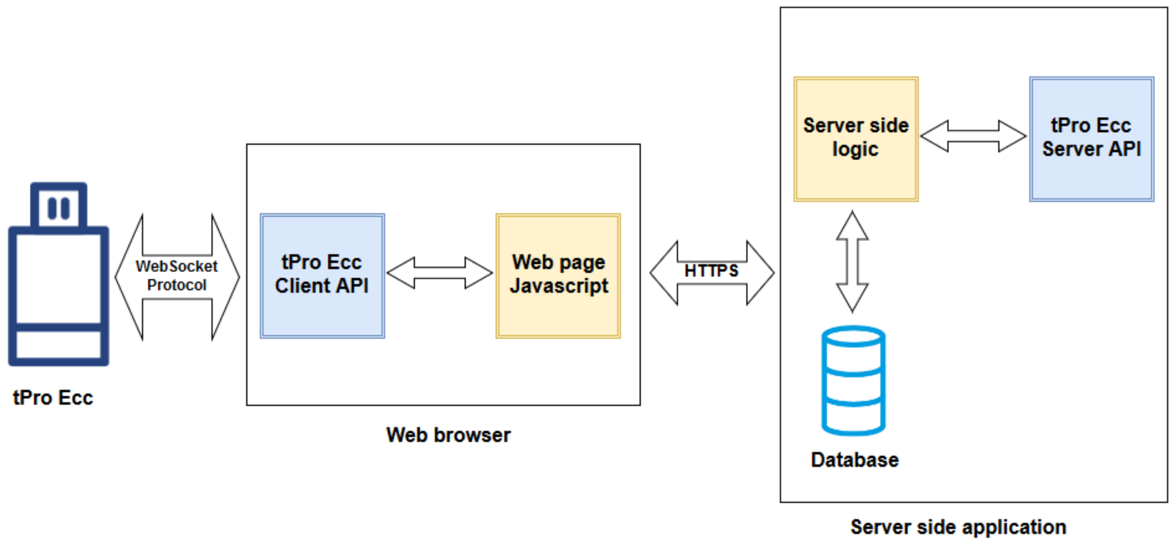

*Figure 2. tPro Ecc client and server API dependency.*

Figure 2. illustrates elements, which constitutes complete ECC token ecosystem. Blue rectangles represents tPro ECC API (both client and server side) provided and maintained by Comarch. Yellow rectangles represents client side scripts and server side logic, which demonstrates usage of specific API methods and are subject to experiments and customization.

**PLEASE NOTE**

Server side API is provided for many different popular web development technology stacks, such as: NodeJS, Java, .Net, Php.

## 2.1. Web browser API

Actual communication with tPro Ecc device is being handled by methods this interface provides. It is up to developer, how to use available functionalities while designing any custom identity and access management system. This API allows developer to trigger tPro Ecc actions, such as:

a) Initialization/deinitialization of the device,
b) Acquiring detailed information concerning token,
c) Generation of ECC key pair,
d) Setting and changing token's PIN,
e) Digital signature of given data,
f) Unblocking device, once it has been blocked.

### 2.1.1. Initialization

Brand new token delivered to the user needs to be initialized before usage in any application. Initialization process involves generation of cryptographic key pair on the token and setup of a new PIN to protect access to token.

Crypto token is intended to improve application security, by increasing trustworthiness that operations confirmed by the token (signed) were done by particular user. In order to be able to account users for their actions, it's necessary to keep a database which stores users and their public keys (generated by token). Database is populated with users' public keys in registration procedure.

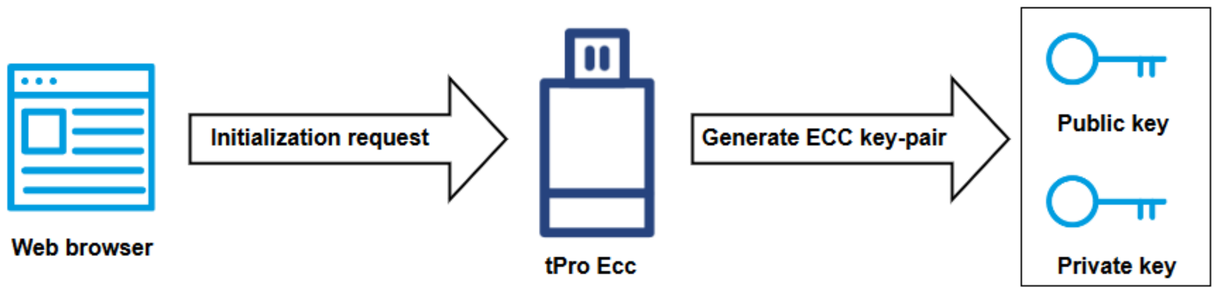

*Figure 3. Initialization diagram.*

### 2.1.2. Registration

Once token is initialized, there is a need to fill up user database with information concerning public key of specific user. Upon successive database update, stored public key is further used for verification of digital signature performed by tPro Ecc token using its private key.

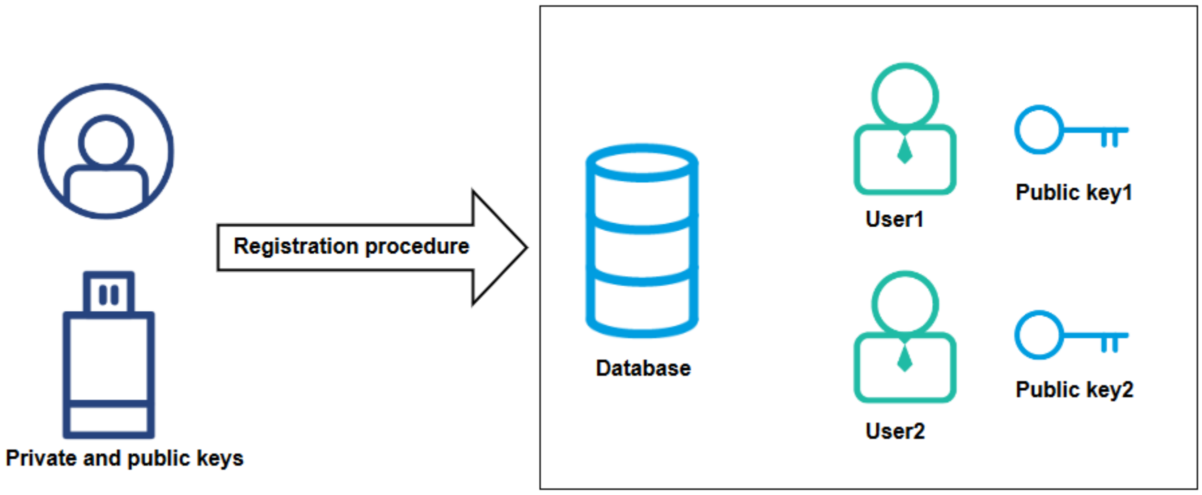

*Figure 4. Registration procedure.*

Initialization and registration (assigning to specific user) of token might be performed altogether. An example of this scenario might contain following steps:

a) User is logged in using existing method (login/password),
b) User sets PIN on the token,
c) Token generates an ECC key pair (private and public key),
d) Public key is returned to the database and stored there for signature verification purposes.

**PLEASE NOTE**

If token is already initialized, re-initialization completely erases content of the token. Keys generated and stored on the token are erased.

**PLEASE NOTE**

It is possible to obtain public key from already initialized token in order to register user in multiple unrelated databases.

### 2.1.3. Generation of ECC key pair

During token initialization process, ECC key pair (public and private key) are being generated, though, it is not the only way to do so. Browser API provides method for generating another key pairs, up to the 4 pairs. While performing other actions by tPro Ecc, user can choose, which key pair should be used for particular action.

What is more, thanks to the possibility of erasure of already stored key pairs, user might freely reconfigure all of the slots.

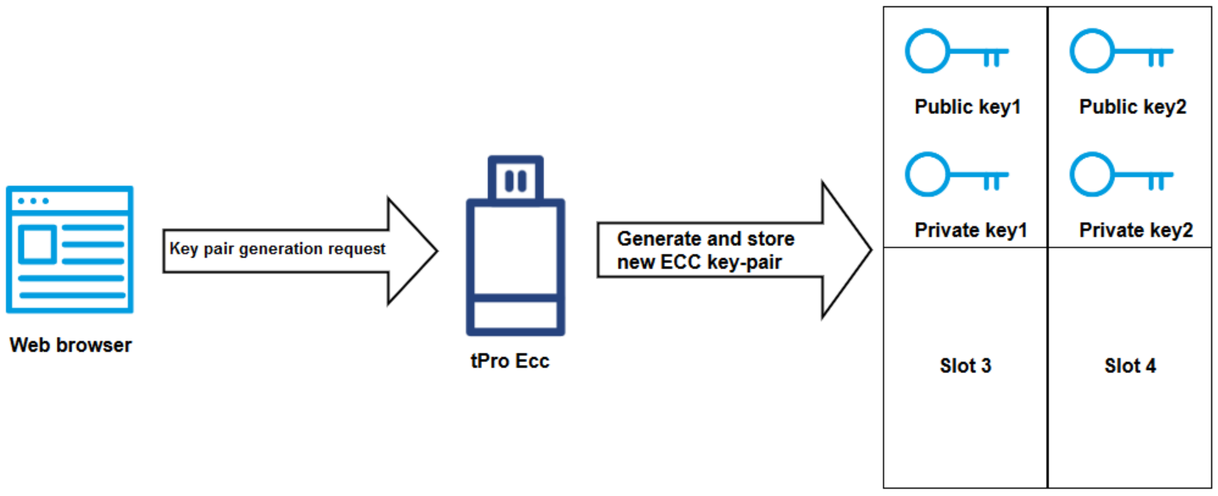

*Figure 5. Generation of new ECC key pair procedure..*

### 2.1.4. Digital signature

This is functionality is the backbone of the whole tPro Ecc device. It can generate digital signature for any data that is being sent to it e.g. random numbers for authentication purposes, information about online banking transfers or PDF files. Web Browser API provides special method to do so. Moreover, user can choose with which of his private keys, stored on token, he wants to sign data.

### 2.1.5. PIN and PUK policy

As shown at *Figure 1.,* upon third failure of entering token's PIN, device becomes blocked and there is no further possibility to use its complex functionalities. It is made to prevent unauthorized use of tPro Ecc. However, there is a way to unblock token using specific method, provided by this API, which requires from a user to know device's PUK code. Upon successful operation token becomes unblocked and user can use him again to perform desired actions.

Each user, for security reasons should periodically, according to the security policy, change the PIN code to their token. For this purpose, the user needs to select the PIN code change method.

Personal Unblocking Key (PUK) of each specific token is being generated during initialization process. It is possible and highly recommended for user to get token's PUK and to store it somewhere safe for future usage. Web Browser API provides specific method for PUK recovery.

### 2.1.6. Detachment

The token might be permanently locked in the system by the administrator in the event of loss, physical damage or the employee dismissal. Thus the token's operation ends.

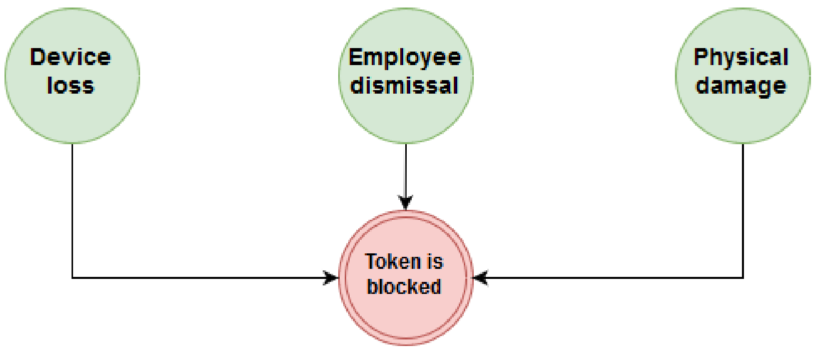

*Figure 6. Detachment of tPro Ecc token.*

## 2.2. Server API

Purpose of this interface existence does not include direct communication with tPro Ecc device. Its main goal is verification of actions performed by token. At this point, developer can check public key validity and digital signature correctness, using proper methods and specific public keys stored in database.

Furthermore, one additional useful method is provided, generation of random number. Developer might send this number via Web Browser API to be signed by tPro Ecc. When device returns digital signature of data sent earlier, server might verify it and use it for authentication purpose.

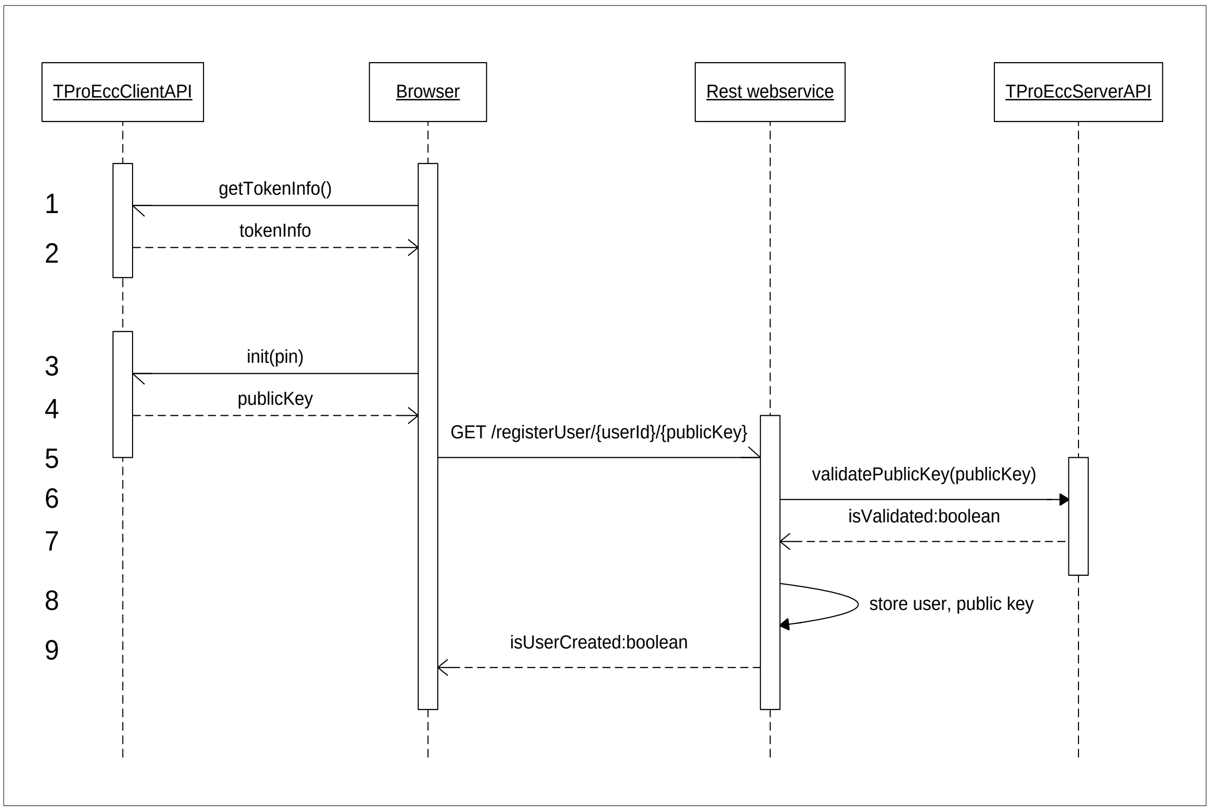

*Figure 7. Server API example flow diagram.*

# 3. Does manufacturer provide any integration tools?

Manufacturer provides javascript based client and server interfaces for triggering actions on the token, as well as for verifying their correctness. What is more, Node.js based integration tutorial for tPro Ecc has been developed. Its main purpose is to improve understanding regarding tPro Ecc token and how to embed it into custom web application.

## 3.1. Client and server side API

Both of these APIs are to be freely used, while integrating custom web application with tPro Ecc device. However, Web browser (Client) API has been split into two interfaces, a simplified one and its full version. Simplified API does contain following methods:

a) *getTokenInfo* - detailed token information are being returned,
b) *init* – initialize token and set its PIN,
c) *sign* – create digital signature of given data,
d) *changePIN* – change token’s PIN,
e) *digest* – create hash of given content.

Full version of API provides the same functionalities as the simplified one, however, with some additions:

a) *generateKey/deleteKey* – generates or deletes ECC key pair,
b) *unblockPin* – used with PUK code, serves to unblock the device,
c) *getPublicKey* – returns chosen public key,
d) *getPUK* – returns token’s PUK,
e) *writeCertificate/getCert* – serves CA certificates management.

Server side API does provide methods, such as:

a) *rand* – generates random number for authentication purposes,
b) *validatePublicKey* – public key verification,
c) *verify signature* – verifies whether tPro Ecc generated signature is valid or not.

## 3.2. Integration tutorial

Following this tutorial will familiarize user with aspects concerning tPro Ecc token, such as:

a) Token management – initialization, registration,
b) Creation of digital signature as well as its verification,
c) Web session authentication.

## 3.3. Direct links

Access tPro Ecc integration tutorial by following link below:

https://github.com/comarch-cybersecurity/tproecc_node_tutorial

Access to documentation for a simplified Web Browser API:

https://github.com/comarch-cybersecurity/tproecc_node_tutorial/tree/master/public_html/doc_Client_Simple

Access to documentation for a full version of Web Browser API:

https://github.com/comarch-cybersecurity/tproecc_node_tutorial/tree/master/public_html/doc_Client

Access to documentation for a Server Node.js API:

https://github.com/comarch-cybersecurity/tproecc_node_tutorial/tree/master/public_html/doc_Server_Node