Scytl

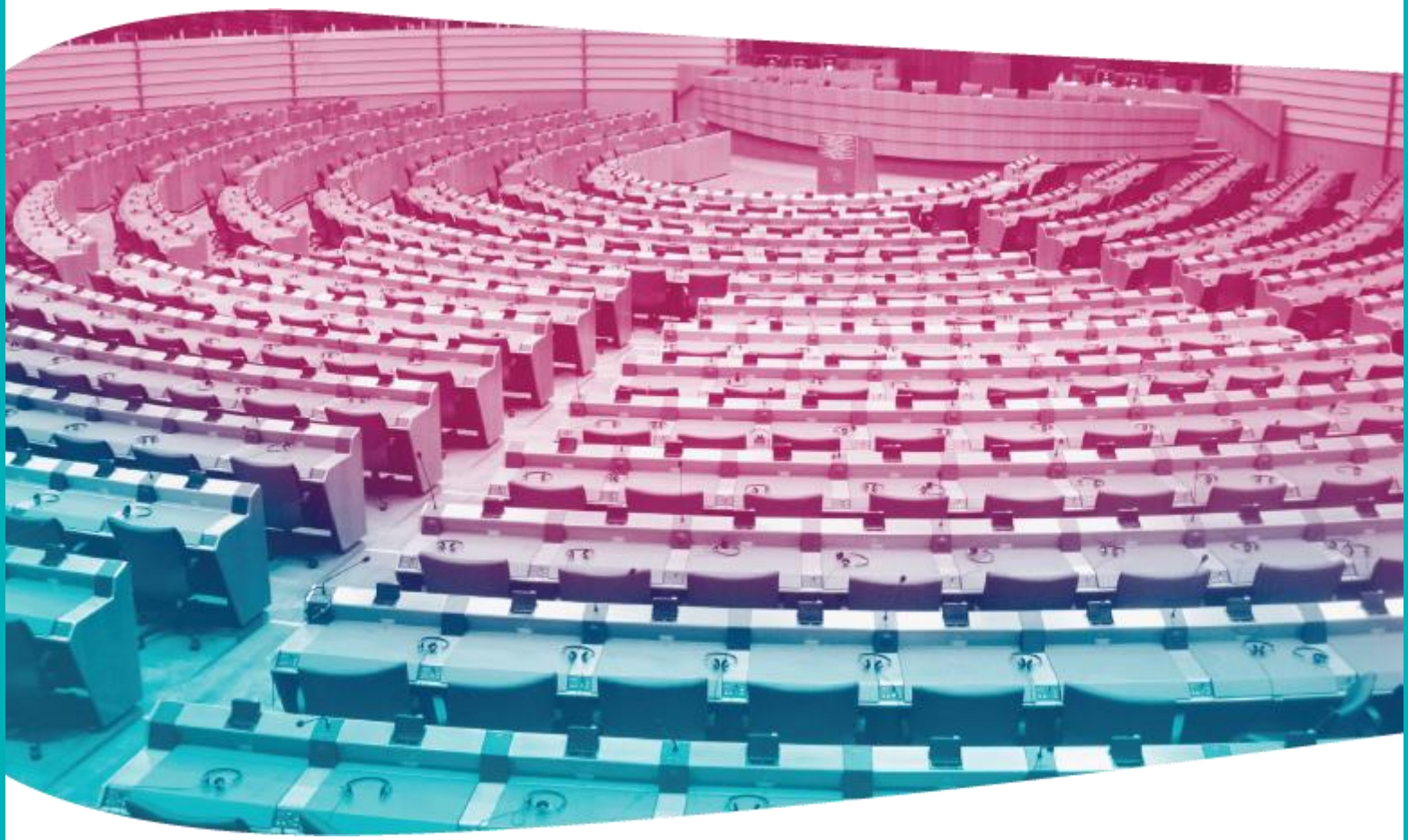
WE POWER DEMOCRACY

Invote

Control Code and Receipt verification - Technical Guide

Software version 4.1.1

Document version 0.1



Scytl – Election Technologies

STRICTLY CONFIDENTIAL

Revision chart

Version	Date	Primary author(s)	Reviewed by	Description
0.1	2022/12/01	NC		Customized version for Basic Counting

Table of contents

01 INTRODUCTION

This document aims to provide information that will allow any third party to validate the ballot box's transparency. More specifically, the information provided herein should allow a third party to create an application that will validate the control code, as well as the receipt against this control code.

01.1 Audience

The intended audience of this document is a software engineer that will undertake the task to create a tool that will validate the control code and the receipt.

Basic understanding of how Invote operates, as well as cryptography are necessary for a meaningful reading of this material.

02 PRELIMINARIES

Before going any further, it is important to understand what the receipt and the control code are used for in Invote.

After casting a vote, the voter is presented with two values: the receipt and the control code (see **Error! Reference source not found.**).

- Receipt: a hash of the Ballot Identifier, a random value computed during the casting of the vote (see section 03.2.1) and encrypted together with the selected voting options. Since the receipt is retrieved during the counting phase, when the vote is decrypted, it is used by the voter to verify that the vote has been successfully decrypted.
- Control Code: a signature of the receipt. It is used to verify the authenticity of the receipt, i.e., to verify that it was generated by the Voting Server.

At the end of the election, the list of receipts is made public, and voters can perform the corresponding verification.

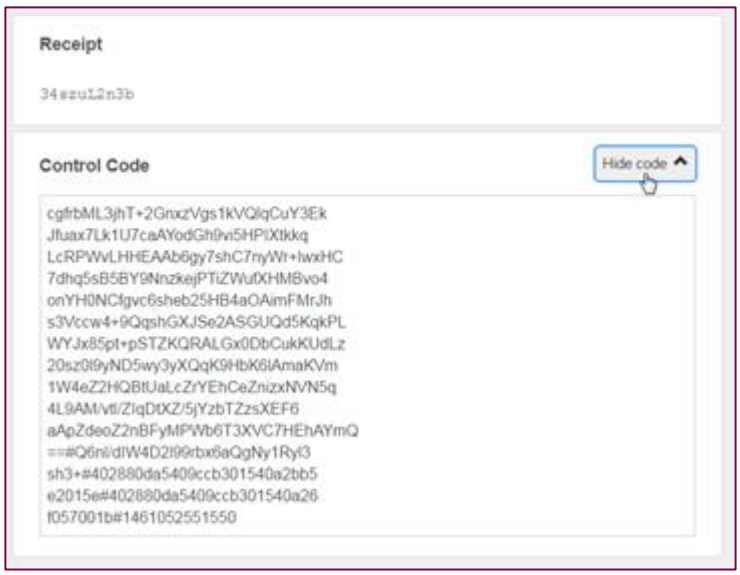


Figure 1 - Receipt and Control code

03 VERIFYING THE RECEIPT AND CONTROL CODE

This section explains how one can verify the receipt that a voter received, as well as the control code.

03.1 Prerequisites

03.1.1 Required data

There are several pieces of information that one must have before validating the control code and the receipt. Namely:

- The message server certificate or public key (only one of the two is necessary)
- The control code itself
- The receipt itself

Once these items are available, we can continue to the next steps.

03.1.2 Cryptographic algorithms

The following cryptographic algorithms are used during the generation and validation of the receipt and the control code.

- Hash function: SHA256
- Signature algorithm: RSA-PSS with a key length of 2048 and SHA256.

03.2 Data structures

03.2.1 The receipt

The receipt is the base64 representation of the hash of the Ballot Identifier (a random value of 128 bits generated when a vote is cast) encoded to an alphanumeric representation (by default Base64 although Base32 can be also used).

```
Receipt = Base64(Hash(BallotIdentifier))
```

For usability purposes, before showing it to the voter, the receipt can be truncated to a given size. The length depends on the number of voters expected in the election and the desired probability of collision.

03.2.2 The control code

The control code is essentially a concatenated string, consisting of five (5) separate pieces, joined by the pound character (#):

- The first part is the base64 representation of the signature of yet another concatenated string (joined together by the semicolon character (;)), consisting of:
  - Base64 representation of the double-hashed ballot identifier
  - Election ID
  - Election Event ID
  - Timestamp of the vote
- The second part is the base64 representation of the ballot identifier.
- The third part is the Election ID.
- The fourth part is the Election Event ID.
- The fifth part is the Timestamp of the vote.

The structure of control code can be found below:

```
Control Code    =
Base64(Signature(Base64(Hash(Hash(BallotIdentifier))) + ";" +
ElectionID  + ";" + ElectionEventID +  + ";" + TimeStamp))
                + "#" + Base64(BallotIdentifier)
                + "#" + ElectionID
                + "#" + ElectionEventID
                + "#" + TimeStamp
```

03.3 Validations

In order to validate the authenticity and the integrity of the receipt, there are two validations that may be made:

- Confirmation that the first part of the control code was indeed signed by the message server and that the control code was not modified.
- Confirmation that the receipt reproduced by the various elements contained in the control code, matches the one provided by the user.

A first step, previous to executing both validations, consists of extracting the five part of the control code, by splitting the provided string on the pound character (#). In pseudocode:

```
control_code_pieces[1] =
Base64(Signature(Base64(Hash(Hash(BallotIdentifier))) + ";" +
ElectionID  + ";" + ElectionEventID +  + ";" + TimeStamp))

control_code_pieces[2] = Base64(BallotIdentifier)

control_code_pieces[3] = electionID

control_code_pieces[4] = electionEventID

control_code_pieces[5] = timestamp
```

03.3.1 Steps to validate the signature

To validate that the first part of the control code was indeed signed by the message server and that the control code has not been modified in any way, one needs to perform the following steps:

1) Reproduce the vote information (VI) as follows:
    - Decode base 64 the Ballot Identifier, stored in control_code_pieces[2]. In pseudocode:
      `ballotId = base64_decode(control_code_pieces[2])`
    - Compute the double hash of the Ballot Identifier and encode it in base64. In pseudocode:
      `VI = base64_encode(hash(hash(ballotId)))`
2) Concatenate the vote information (VI) obtained in the previous step, with the values stored in the fields control_code_pieces[3], control_code_pieces[4] and control_code_pieces[5], using as a separator the semicolon character. This is the signed data (SD). In pseudocode:
   `SD=VI;electionID;electionEventID;timestamp`
3) Decode base64 the signature, stored in control_code_pieces[1]. In pseudocode:
   `signature = base64_decode(control_code_pieces[2])`
4) Validate that the signature obtained in the previous is correct using the message server's public key and the signed data (SD).
   `sign_verify(message_server_pk,signature,SD)`

03.3.2 Steps to validate the receipt

To validate the receipt, one need to execute the following steps:

1) Reproduce the receipts (RR) as follows:

- Decode base 64 the Ballot Identifier, stored in control_code_pieces[2]. In pseudocode:

```
ballotId = base64_decode(control_code_pieces[2])
```

- Compute the hash of the Ballot Identifier and encode it in base64. In pseudocode:

```
VI = base64_encode(hash(ballotId))
```

2) Check that the receipt provided matches the X first characters of the reproduced receipt (RR).

03.4 Expected output

If both the validations are successful, the tool should output a success message. In any other case, it should throw an error, indicating what went wrong. Indicatively:

- If one or more of the required pieces is missing
- If any item provided is invalid (for example if the control code does not have the expected structure)
- If any of the validations fail