

# CHVote – Sixteen Best Practices and Lessons Learned

*R. Haenni, E. Dubuis, R. E. Koenig, P. Locher*  
E-Vote-ID 2020, October 8th, 2020

# Paper Overview

- Item 1: Modeling the Electoral Systems
- Item 2: Modeling the Electorate
- Item 3: Cryptographic Building Blocks
- Item 4: Cryptographic Parameters
- Item 5: Parties and Communication
- Item 6: Protocol Structure and Communication Diagrams
- Item 7: Pseudo-Code Algorithms
- Item 8: Usability and Performance
- Item 9: Mathematical Library
- Item 10: Naming Conventions
- Item 11: Implementation of Pseudo-Code Algorithms
- Item 12: Parameter Validity Checks
- Item 13: Implementation of Protocol Parties
- Item 14: Cryptographically Relevant Code
- Item 15: Transparency
- Item 16: Verifier

# Paper Overview

- Item 1: Modeling the Electoral Systems
- Item 2: Modeling the Electorate
- Item 3: Cryptographic Building Blocks
- Item 4: Cryptographic Parameters
- Item 5: Parties and Communication
- Item 6: Protocol Structure and Communication Diagrams
- Item 7: Pseudo-Code Algorithms
- Item 8: Usability and Performance
- Item 9: Mathematical Library
- Item 10: Naming Conventions
- Item 11: Implementation of Pseudo-Code Algorithms
- Item 12: Parameter Validity Checks
- Item 13: Implementation of Protocol Parties
- Item 14: Cryptographically Relevant Code
- Item 15: Transparency
- Item 16: Verifier

Joshua Bloch

Updated  
for  
Java 9



# Effective Java

## Third Edition

Best practices for



...the Java Platform



## Contents

<b>Foreword</b> .....	<b>xi</b>
<b>Preface</b> .....	<b>xiii</b>
<b>Acknowledgments</b> .....	<b>xvii</b>
<b>1 Introduction</b> .....	<b>1</b>
<b>2 Creating and Destroying Objects</b> .....	<b>5</b>
Item 1: Consider static factory methods instead of constructors ...	5
Item 2: Consider a builder when faced with many constructor parameters .....	10
Item 3: Enforce the singleton property with a private constructor or an enum type .....	17
Item 4: Enforce noninstantiability with a private constructor ...	19
Item 5: Prefer dependency injection to hardwiring resources ...	20
Item 6: Avoid creating unnecessary objects .....	22
Item 7: Eliminate obsolete object references .....	26
Item 8: Avoid finalizers and cleaners .....	29
Item 9: Prefer try-with-resources to try-finally .....	34
<b>3 Methods Common to All Objects</b> .....	<b>37</b>
Item 10: Obey the general contract when overriding equals .....	37
Item 11: Always override hashCode when you override equals ...	50
Item 12: Always override toString .....	55
Item 13: Override clone judiciously .....	58
Item 14: Consider implementing Comparable .....	66
<b>4 Classes and Interfaces</b> .....	<b>73</b>
Item 15: Minimize the accessibility of classes and members ...	73
Item 16: In public classes, use accessor methods, not public fields	78
Item 17: Minimize mutability .....	80
Item 18: Favor composition over inheritance .....	87

vii



# Item 17: Software Announcement

# Item 17: Software Announcement

To continue the CHVote project independently of the support from the State of Geneva, a new funding from *eGovernment Switzerland* has been acquired by the Bern University of Applied Sciences in August 2019. The main goal of this project was to release a final stable version of the specification document and to update the cryptographic core of the protocol based on the code released by the State of Geneva. As a first project deliverable, the current Version 3.0 of the specification document has been released in December 2019 [9]. **At the time of writing this paper, the developed *OpenCHVote* software is not yet complete. Since the project is in its final stage, the code is expected to be released soon under a non-proprietary license.**<sup>3</sup> The general purpose of the project is to make the achievements available to others for pursuing it further.

---

<sup>1</sup> For further details about the reasons for abandoning the project, we refer to the State Council's press statement at <https://www.ge.ch/document/12832/telecharger>.

<sup>2</sup> See <https://chvote2.gitlab.io>

<sup>3</sup> See <https://gitlab.com/openchvote>

# Item 17: Software Announcement

To continue the CHVote project independently of the support from the State of Geneva, a new funding from *eGovernment Switzerland* has been acquired by the Bern University of Applied Sciences in August 2019. The main goal of this project was to release a final stable version of the specification document and to update the cryptographic core of the protocol based on the code released by the State of Geneva. As a first project deliverable, the current Version 3.0 of the specification document has been released in December 2019 [9]. **At the time of writing this paper, the developed *OpenCHVote* software is not yet complete. Since the project is in its final stage, the code is expected to be released soon under a non-proprietary license.**<sup>3</sup> The general purpose of the project is to make the achievements available to others for pursuing it further.

---

<sup>1</sup> For further details about the reasons for abandoning the project, we refer to the State Council's press statement at <https://www.ge.ch/document/12832/telecharger>.

<sup>2</sup> See <https://chvote2.gitlab.io>

<sup>3</sup> See <https://gitlab.com/openchvote>

## Lesson Learned

Don't announce releasing software before you are ready to release it

# Item 17: Software Announcement

but . . .



# Item 17: Software Announcement

- ▶ **OpenCHVote v1.0** has been released yesterday on  
<https://gitlab.com/openchvote>
- ▶ Along with an updated **CHVote Protocol Specification v3.1**  
<https://eprint.iacr.org/2017/325>
- ▶ GNU Affero General Public License Version 3
- ▶ Funded by the State of Geneva (2016–2019) and the Federal Chancellery (2019–2020)

# Item 17: Software Announcement

## Key Features

- ▶ Complete parameterizable CHVote protocol run
- ▶ Simultaneous execution of multiple protocol runs
- ▶ Two protocol variants: *plain* and *writein*
- ▶ Optimized performance
- ▶ Clean infrastructure interfaces to cryptographically irrelevant components
- ▶ No compulsory dependencies to third-party libraries/frameworks
- ▶ Straightforward installation

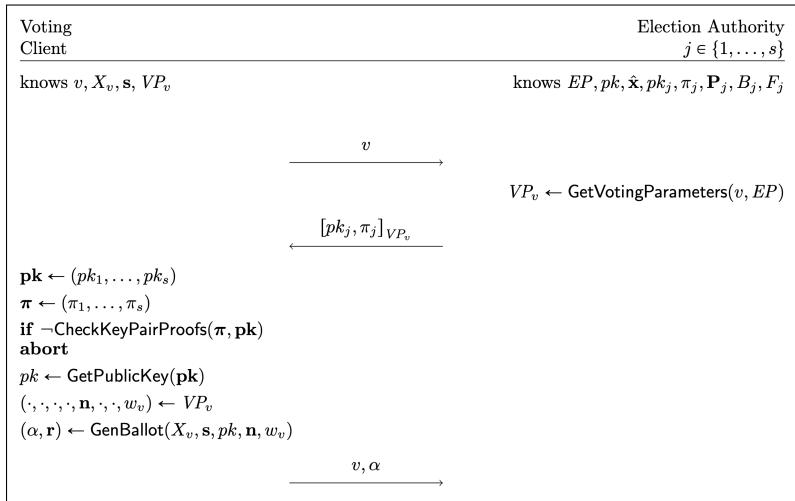
# Item 17: Software Announcement

## Missing Features

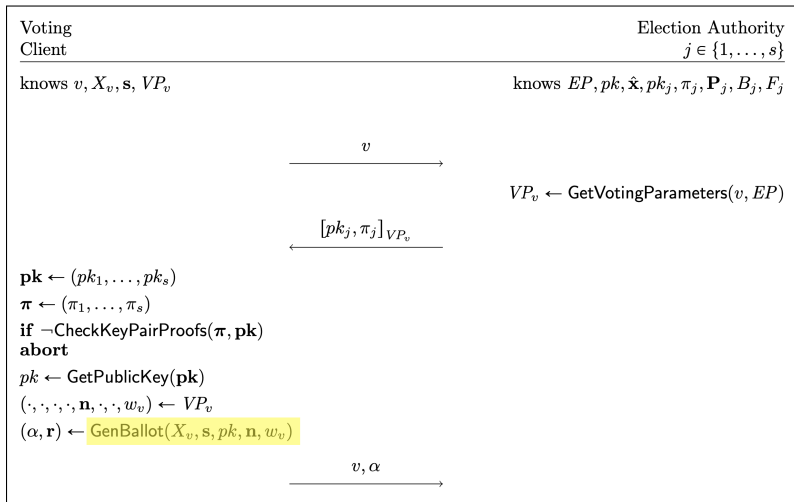
- ▶ Software for performing universal verification (the Verifier)
- ▶ JavaScript implementation of the voting client
- ▶ Reliable and secure infrastructure services
- ▶ Other infrastructure software components

# Items 7 & 11: Pseudo-Code Algorithms

# Item 7: Pseudo-Code Algorithms



# Item 7: Pseudo-Code Algorithms



# Item 7: Pseudo-Code Algorithms

**Algorithm:** GenBallot( $X, \mathbf{s}, pk, \mathbf{n}, w$ )

**Input:** Voting code  $X \in A_X^{\ell_X}$

Selection  $\mathbf{s} = (s_1, \dots, s_k), 1 \leq s_1 < \dots < s_k \leq n$

Encryption key  $pk \in \mathbb{G}_q$

Number of candidates  $\mathbf{n} = (n_1, \dots, n_t), n_j \in \mathbb{N}^+, n = \sum_{j=1}^t n_j$

Counting circle  $w \in \mathbb{N}^+$

$x \leftarrow \text{ToInteger}(X, A_x)$  // see Alg. 4.8

$\hat{x} \leftarrow \hat{g}^x \bmod \hat{p}$

$\mathbf{p} \leftarrow \text{GetPrimes}(n + w)$  //  $\mathbf{p} = (p_0, \dots, p_{n+w})$ , see Alg. 8.1

$\mathbf{m} \leftarrow \text{GetEncodedSelections}(\mathbf{s}, \mathbf{p})$  //  $\mathbf{m} = (m_1, \dots, m_k)$ , see Alg. 8.24

$m \leftarrow \prod_{j=1}^k m_j$

**if**  $p_{n+w} \cdot m \geq p$  **then**

**return**  $\perp$  //  $\mathbf{s}, \mathbf{n}$ , and  $w$  are incompatible with  $p$

$(\mathbf{a}, \mathbf{r}) \leftarrow \text{GenQuery}(\mathbf{m}, pk)$  //  $\mathbf{a} = (a_1, \dots, a_k), \mathbf{r} = (r_1, \dots, r_k)$ , see Alg. 8.25

$r \leftarrow \sum_{j=1}^k r_j \bmod q$

$\pi \leftarrow \text{GenBallotProof}(x, m, r, \hat{x}, \mathbf{a}, pk)$  // see Alg. 8.26

$\alpha \leftarrow (\hat{x}, \mathbf{a}, \pi)$

**return**  $(\alpha, \mathbf{r})$  //  $\alpha \in \mathbb{G}_{\hat{q}} \times (\mathbb{G}_q^2)^k \times (\mathbb{Z}_{2^{\tau}} \times (\mathbb{Z}_{\hat{q}} \times \mathbb{G}_q \times \mathbb{Z}_q))$ ,  $\mathbf{r} \in \mathbb{Z}_q^k$

# Item 7: Pseudo-Code Algorithms

**Algorithm:** GenBallot( $X, \mathbf{s}, pk, \mathbf{n}, w$ )

**Input:** Voting code  $X \in A_X^{\ell_X}$

Selection  $\mathbf{s} = (s_1, \dots, s_k), 1 \leq s_1 < \dots < s_k \leq n$

Encryption key  $pk \in \mathbb{G}_q$

Number of candidates  $\mathbf{n} = (n_1, \dots, n_t), n_j \in \mathbb{N}^+, n = \sum_{j=1}^t n_j$

Counting circle  $w \in \mathbb{N}^+$

$x \leftarrow \text{ToInteger}(X, A_x)$

// see Alg. 4.8

$\hat{x} \leftarrow \hat{g}^x \bmod \hat{p}$

$\mathbf{p} \leftarrow \text{GetPrimes}(n + w)$

//  $\mathbf{p} = (p_0, \dots, p_{n+w})$ , see Alg. 8.1

$\mathbf{m} \leftarrow \text{GetEncodedSelections}(\mathbf{s}, \mathbf{p})$

//  $\mathbf{m} = (m_1, \dots, m_k)$ , see Alg. 8.24

$m \leftarrow \prod_{j=1}^k m_j$

**if**  $p_{n+w} \cdot m \geq p$  **then**

**return**  $\perp$

//  $\mathbf{s}, \mathbf{n}$ , and  $w$  are incompatible with  $p$

$(\mathbf{a}, \mathbf{r}) \leftarrow \text{GenQuery}(\mathbf{m}, pk)$

//  $\mathbf{a} = (a_1, \dots, a_k), \mathbf{r} = (r_1, \dots, r_k)$ , see Alg. 8.25

$r \leftarrow \sum_{j=1}^k r_j \bmod q$

$\pi \leftarrow \text{GenBallotProof}(x, m, r, \hat{x}, \mathbf{a}, pk)$

// see Alg. 8.26

$\alpha \leftarrow (\hat{x}, \mathbf{a}, \pi)$

**return**  $(\alpha, \mathbf{r})$

//  $\alpha \in \mathbb{G}_{\hat{q}} \times (\mathbb{G}_{\hat{q}}^2)^k \times (\mathbb{Z}_{2\tau} \times (\mathbb{Z}_{\hat{q}} \times \mathbb{G}_q \times \mathbb{Z}_q))$ ,  $\mathbf{r} \in \mathbb{Z}_q^k$



# Item 11: Implementation of Algorithms

```
public class GenBallot {

    public static Pair<Ballot, Vector<BigInteger>>
run(String X, IntVector bold_s, QuadraticResidue pk, IntVector bold_n, int w, Parameters params)

    // PREPARATION
    Precondition.checkNotNull(X, bold_s, pk, bold_n, params);
    int k = bold_s.getLength();
    int t = bold_n.getLength();
    int n = Math.intSum(bold_n);
    Precondition.check(params.GG_q.contains(pk));
    Precondition.check(IntSet.NW_plus.contains(w));
    Precondition.check(Set.String(params.A_X, params.ell_X).contains(X));
    Precondition.check(Set.IntVector(IntSet.NW_plus, t).contains(bold_n));
    Precondition.check(Set.IntVector(IntSet.NW_plus(n), k).contains(bold_s));
    Precondition.check(bold_s.isSorted());

    // ALGORITHM
    var x = ToInteger.run(X, params.A_X);
    var x_hat = Mod.pow(params.g_hat, x, params.p_hat);
    var bold_p = GetPrimes.run(n + w, params);
    var bold_m = GetEncodedSelections.run(bold_s, bold_p);
    var m = Math.prod(bold_m.map(QuadraticResidue::getValue));
    if (bold_p.getValue(n + w).getValue().multiply(m).compareTo(params.p) >= 0)
        throw new AlgorithmException(GenBallot.class, AlgorithmException.Type.INCOMPATIBLE_MATRIX);
    var pair = GenQuery.run(bold_m, pk, params);
    var bold_a = pair.getFirst();
    var bold_r = pair.getSecond();
    var r = Mod.sum(bold_r, params.q);
    var pi = GenBallotProof.run(x, Mod.prod(bold_m), r, x_hat, bold_a, pk, params);
    var alpha = new Ballot(x_hat, bold_a, pi);
    return new Pair<>(alpha, bold_r);
}
}
```

# Item 11: Implementation of Algorithms

```
public class GenBallot {  
  
    public static Pair<Ballot, Vector<BigInteger>>  
run(String X, IntVector bold_s, QuadraticResidue pk, IntVector bold_n, int w, Parameters params)  
  
    // PREPARATION  
    Precondition.checkNotNull(X, bold_s, pk, bold_n, params);  
    int k = bold_s.getLength();  
    int t = bold_n.getLength();  
    int n = Math.intSum(bold_n);  
    Precondition.check(params.GG_q.contains(pk));  
    Precondition.check(IntSet.NW_plus.contains(w));  
    Precondition.check(Set.String(params.A_X, params.ell_X).contains(X));  
    Precondition.check(Set.IntVector(IntSet.NW_plus, t).contains(bold_n));  
    Precondition.check(Set.IntVector(IntSet.NW_plus(n), k).contains(bold_s));  
    Precondition.check(bold_s.isSorted());  
  
    // ALGORITHM  
    var x = ToInteger.run(X, params.A_X);  
    var x_hat = Mod.pow(params.g_hat, x, params.p_hat);  
    var bold_p = GetPrimes.run(n + w, params);  
    var bold_m = GetEncodedSelections.run(bold_s, bold_p);  
    var m = Math.prod(bold_m.map(QuadraticResidue::getValue));  
    if (bold_p.getValue(n + w).getValue().multiply(m).compareTo(params.p) >= 0)  
        throw new AlgorithmException(GenBallot.class, AlgorithmException.Type.INCOMPATIBLE_MATRIX);  
    var pair = GenQuery.run(bold_m, pk, params);  
    var bold_a = pair.getFirst();  
    var bold_r = pair.getSecond();  
    var r = Mod.sum(bold_r, params.q);  
    var pi = GenBallotProof.run(x, Mod.prod(bold_m), r, x_hat, bold_a, pk, params);  
    var alpha = new Ballot(x_hat, bold_a, pi);  
    return new Pair<>(alpha, bold_r);  
}  
}
```

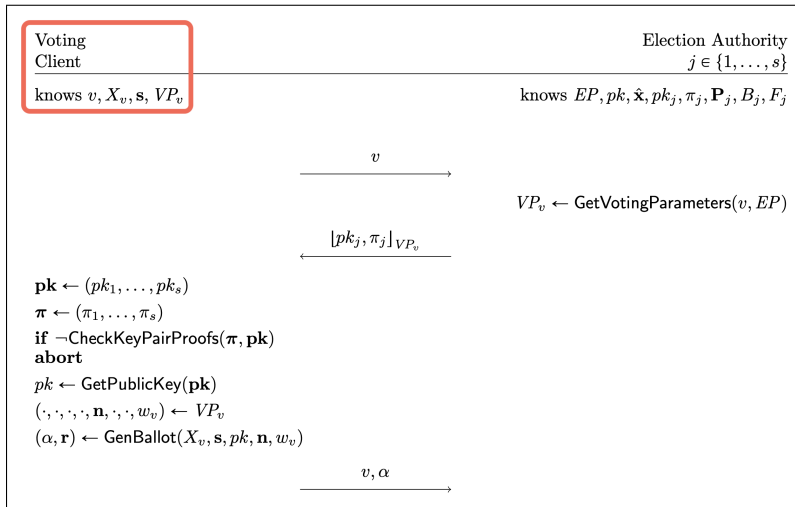
# Item 11: Implementation of Algorithms

$$\begin{aligned}x &\leftarrow \text{ToInteger}(X, A_x) \\ \hat{x} &\leftarrow \hat{g}^x \bmod \hat{p} \\ \mathbf{p} &\leftarrow \text{GetPrimes}(n + w) \\ \mathbf{m} &\leftarrow \text{GetEncodedSelections}(\mathbf{s}, \mathbf{p}) \\ m &\leftarrow \prod_{j=1}^k m_j\end{aligned}$$

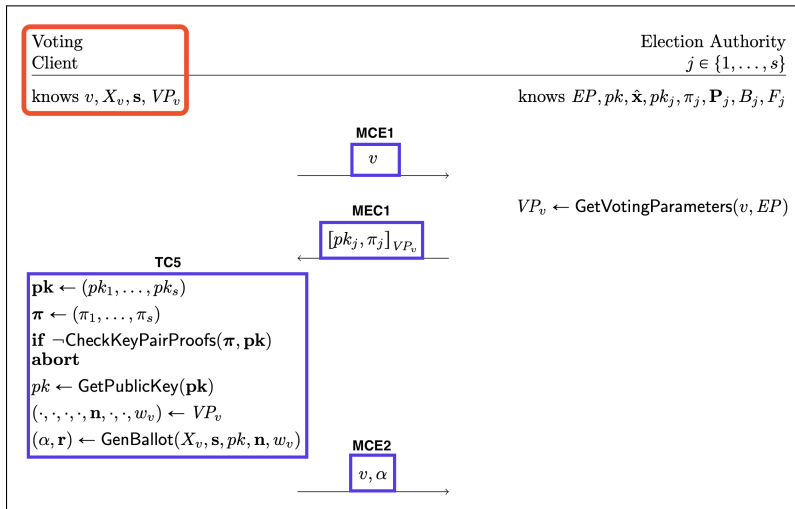
```
var x = ToInteger.run(X, params.A_X);  
var x_hat = Mod.pow(params.g_hat, x, params.p_hat);  
var bold_p = GetPrimes.run(n + w, params);  
var bold_m = GetEncodedSelections.run(bold_s, bold_p);  
var m = Math.prod(bold_m.map(QuadraticResidue::getValue));
```

# Item 13: Protocol Parties

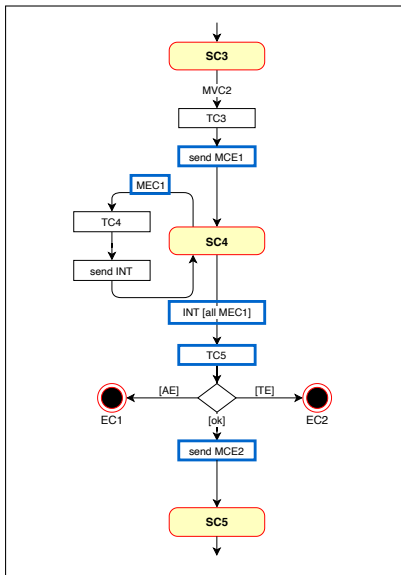
# Item 13: Protocol Parties



# Item 13: Protocol Parties



# Item 13: Protocol Parties



# Item 13: Protocol Parties

```
80 //  
81 @Override  
82 public void handleInternalMessage(EventContext context) {  
83     var eventSetup = context.getEventSetup();  
84     var eventMessages = context.getEventMessages();  
85     var eventData = (EventData) context.getEventData();  
86  
87     // check if all MEC1 messages are available  
88     if (eventMessages.hasAllMessages(eventSetup, MessageType.MEC1)) {  
89         var params = new Parameters(eventSetup.getSecurityLevel());  
90         try {  
91             // perform task  
92             TC5.run(eventData, params);  
93             // select event data  
94             var v = eventData.get_v();  
95             var alpha = eventData.get_alpha();  
96             // send MCE2 message to all election authorities  
97             this.party.sendMessage(new MCE2(v, alpha), eventSetup);  
98             // update state  
99             context.setCurrentState(SC5.class);  
100         } catch (AlgorithmException exception) {  
101             // move to error state  
102             context.setCurrentState(EC1.class);  
103         } catch (TaskException exception) {  
104             // move to error state  
105             context.setCurrentState(EC2.class);  
106         }  
107     }  
108 }
```



# Item 13: Protocol Parties

```
public class TC5 {  
3   public static void run(EventData eventData, Parameters params) {  
        // select event data  
        var X_v = eventData.get_X_v();  
        var bold_s = eventData.get_bold_s();  
        var VP_v = eventData.get_VP_v();  
  
        // perform main task  
        var bold_pk = eventData.get_bold_pk();  
        var bold_pi = eventData.get_bold_pi();  
3       if (!CheckKeyPairProofs.run(bold_pi, bold_pk, params)) {  
            throw new TaskException(TC5.class, TaskException.Type.INVALID_ZKP_PROOF);  
3       }  
        var pk = GetPublicKey.run(bold_pk, params);  
        var bold_n = VP_v.get_bold_n();  
        var w_v = VP_v.get_w_v();  
        var pair = GenBallot.run(X_v, bold_s, pk, bold_n, w_v, params);  
        var alpha = pair.getFirst();  
        var bold_r = pair.getSecond();  
  
        // update event data  
        eventData.set_pk(pk);  
        eventData.set_alpha(alpha);  
        eventData.set_bold_r(bold_r);  
3   }  
}
```

# Item 13: Protocol Parties

```
pk ← ( $pk_1, \dots, pk_s$ )  
 $\pi$  ← ( $\pi_1, \dots, \pi_s$ )  
if ¬CheckKeyPairProofs( $\pi$ , pk)  
abort  
pk ← GetPublicKey(pk)  
( $\cdot, \cdot, \cdot, \cdot, \mathbf{n}, \cdot, \cdot, w_v$ ) ←  $VP_v$   
( $\alpha$ , r) ← GenBallot( $X_v, \mathbf{s}, pk, \mathbf{n}, w_v$ )
```

```
var bold_pk = eventData.get_bold_pk();  
var bold_pi = eventData.get_bold_pi();  
if (!CheckKeyPairProofs.run(bold_pi, bold_pk, params))  
    throw new TaskException(TCS.class, TaskException.Type.INVALID_ZKP_PROOF);  
var pk = GetPublicKey.run(bold_pk, params);  
var bold_n = VP_v.get_bold_n();  
var w_v = VP_v.get_w_v();  
var pair = GenBallot.run(X_v, bold_s, pk, bold_n, w_v, params);  
var alpha = pair.getFirst();  
var bold_r = pair.getSecond();
```

# Conclusion

# Conclusion

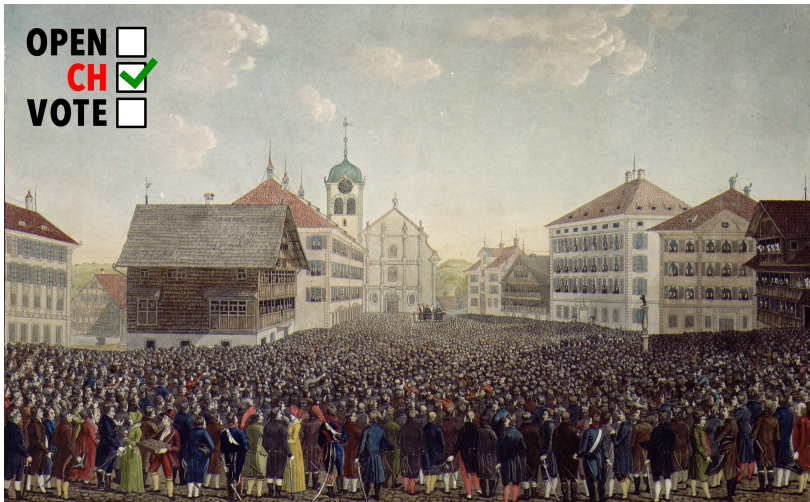
## Achievements

- ▶ CHVote: Consistent and comprehensive protocol specification
- ▶ OpenCHVote: Perfectly matching Java implementation
- ▶ Cryptographically relevant aspects fully covered
- ▶ Flexible software design
- ▶ Clean production-quality code

## Next Steps

- ▶ Public scrutiny
- ▶ Integration of third-party contributions
- ▶ Collaborations with third parties on extensions and open issues

# Questions?



Source: <https://en.wikipedia.org/wiki/Landsgemeinde>