Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

Quelle: computerworld.ch

Sicherheit bei «e-Voting»
Privatim-Anlass 25. September 2018, Zürich

Prof. Dr. Eric Dubuis

- Research Institute for Security in the Information Society

Zu mir

- Prof. Dr. Eric Dubuis
- Departement Technik und Informatik
- Leiter des Research Institute for Security in the Information Society (RISIS)
- Mitbegründer des Swiss E-Voting Competence Center
- Forschung zu elektronischen Wahlen übers Internet

Inhalt

- Einleitung
- Grundlagen
- Wahrung des Stimmgeheimnisses
- Von der Theorie in die Praxis
- Zusammenfassung

Einleitung

Was soll uns ein Wahlsystem garantieren?

«Demokratie»-Regeln:

- Nur Stimmen von Stimmberechtigten fliessen ins Resultat ein («Berechtigung»)
- Eine stimmberechtigte Person ←→ eine Stimme («Eine-Stimme-Eigenschaft»)
- Das Resultat ist erst nach Urnenschluss bekannt («Fairness»)

Regeln zum Schutz der Privatsphäre:

- Die Stimme ist geheim («Stimmgeheimnis»)
- Abstimmende Person kann nicht beweisen, wie sie gestimmt hat («keine Quittung»)

Verifizierbarkeit:

- Wurde meine Stimme richtig gezählt? («individuelle Verifizierbarkeit»)
- Wurden alle gültigen Stimmen gezählt? («universelle Verifizierbarkeit»)

Herausforderungen

Entwicklung eines E-Voting-Systems mit teilweise widersprüchlichen Anforderungen:

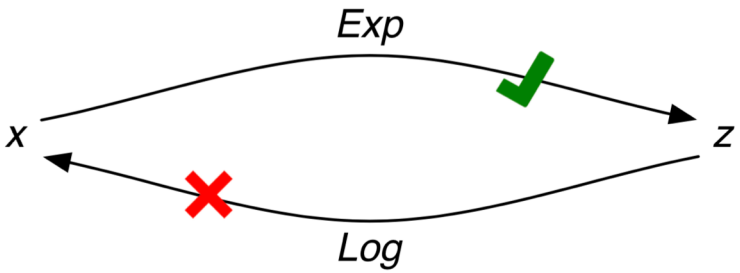
- Verifizierbarkeit ←→ Wahlgeheimnis
- Anonymität ←→ Wahlberechtigung

Grundlagen

Ein paar Grundlagen

Einweg-Funktion (in grossen und begrenzten Zahlenräumen)

- $z = Exp(x)$ ist leicht
- $x = Log(z)$ ist sehr schwierig



Homomorphe Verschlüsselung

- $Encrypt(x) * Encrypt(y) = Encrypt(x + y)$
- $Encrypt(x)^y = Encrypt(x \cdot y)$

Kryptografische Beweise

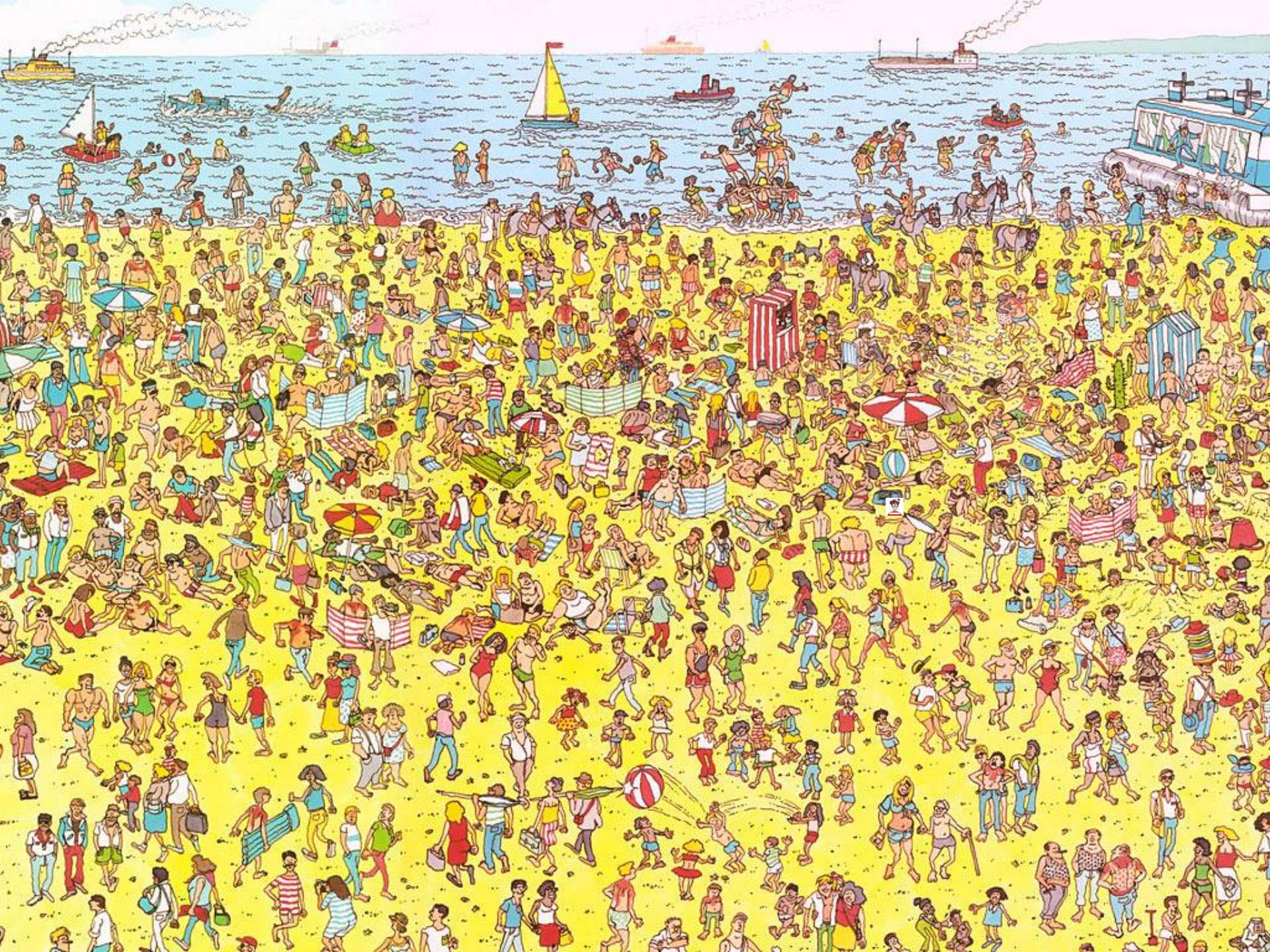
- Wie kann ich beweisen, dass ich den Wert *x* kenne, so dass *z = Exp(x)* gilt?

- Varianten:

  1. *x* offen legen

  2. Erstellen eines kryptographischen Beweises
     - Wähle *r* zufällig
     - Berechne *t = Exp(r)*
     - Berechne *c = hash(t,z)*
     - Berechne $s = r + x \cdot c$
     - **Veröffentlichung von *(t,c,s)***

     Der Beweis ist gültig, falls $Exp(s) = t \cdot z^c$ und $c = hash(t, z)$



Anwendung kryptografischer Beweise

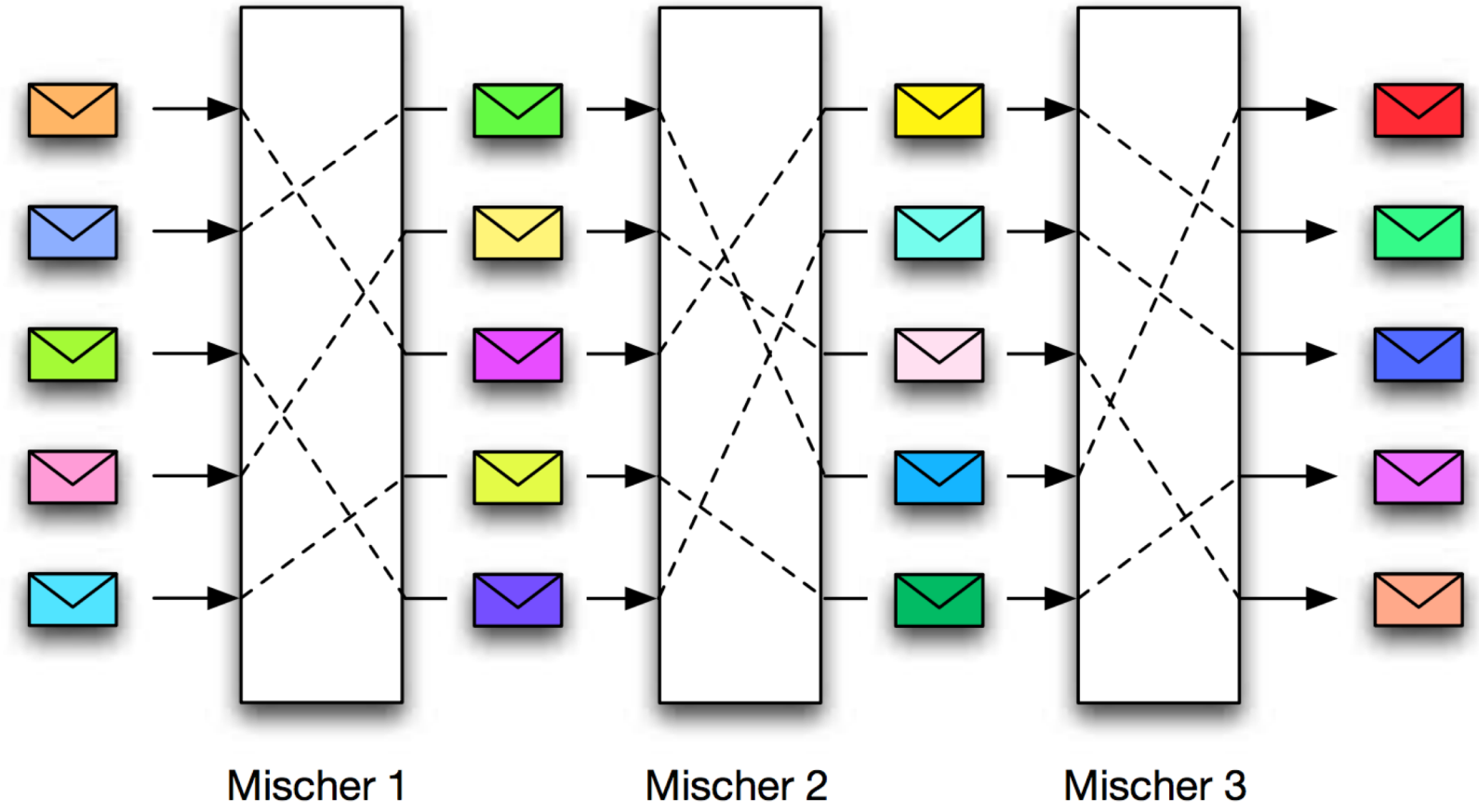
- Verschlüsselung der Stimme
  - Beweisen, dass die verschlüsselte Stimme 0 oder 1 ist
- Zusammenzählen von verschlüsselten Stimmen (ohne sie zu entschlüsseln)
- Berechnen von Prüfcodes ohne die Stimmen zu entschlüsseln
- Wiederverschlüsselung von verschlüsselten Stimmen
- Kryptographisches Mischen von verschlüsselten Stimmen
  - Beweisen, dass korrekt gemischt wurde
- Teilen des privaten Schlüssels für die Entschlüsselung
- Entschlüsselung mittels privaten Teilschlüsseln
- etc.

Wahrung des Stimmgeheimnisses

Entkopplung der Identität von der Stimmen

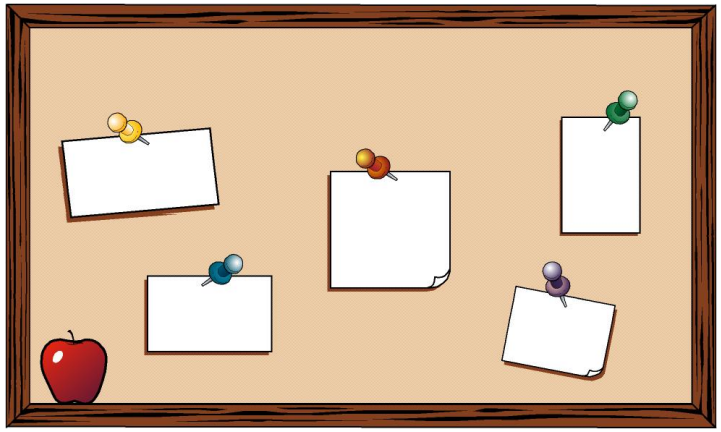
- Blinde Signaturen und anonymer Kanal
  - Eine Meldung m wird von Autorität *X* signiert, ohne dass *X* den Inhalt von *m* kennt
  - Die von *X* signierte Meldung *m* wird über einen anonymen Kanal abgegeben
- Erstellen von anonymen Berechtigungen
  - Aus eine Schlüsselpaar *(sk, pk)* wird ein neuer, zu *sk* passender öffentlicher Schlüssel *pk'* erzeugt
  - Die Meldung *m* wird mit *sk* signiert
  - Die mit *sk* signierte Meldung *m* wird über einen anonymen Kanal abgegeben
- Mit kryptografischem Mix-Net

Illustration eines kryptografischen Mix-Net



Das elektronische Anschlagbrett

Publiziert Daten wie:



- Kryptografische Parameter
- Digitale Zertifikate mit öffentlichen Schlüsseln
- Wahldaten wie Kandidierende und Wahlberechtigte, Wahlregeln
- Elektorat
- Eingetroffene, verschlüsselte und identifizierbare Stimmen (mit Beweisen)
- Die verschlüsselten Stimmen nach der Entkopplung durch ein kryptografisches Mischnetzwerk (mit Beweisen)
- Die entschlüsselten Stimmen (mit Beweisen)
- Das Resultat

Das elektronische Anschlagbrett muss «fälschungssicher» sein!

Von der Theorie in die Praxis

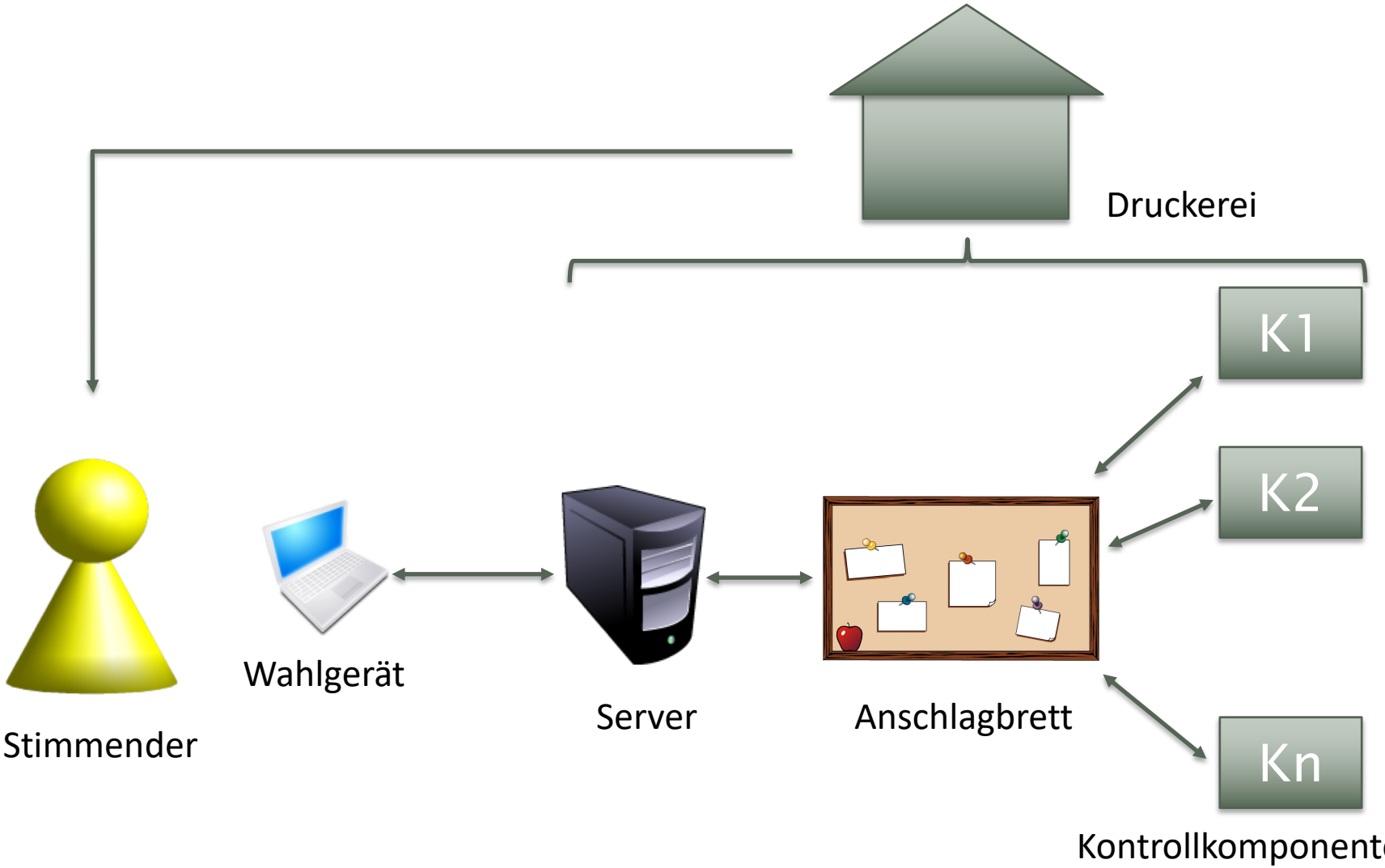
Verordnung «Elektronische Stimmabgabe»

Verordnung der BK über die elektronische Stimmabgabe (VEleS) vom 13. Dezember 2013 (Stand am 1. Juli 2018)

Umfassende Sicherheitsanforderungen

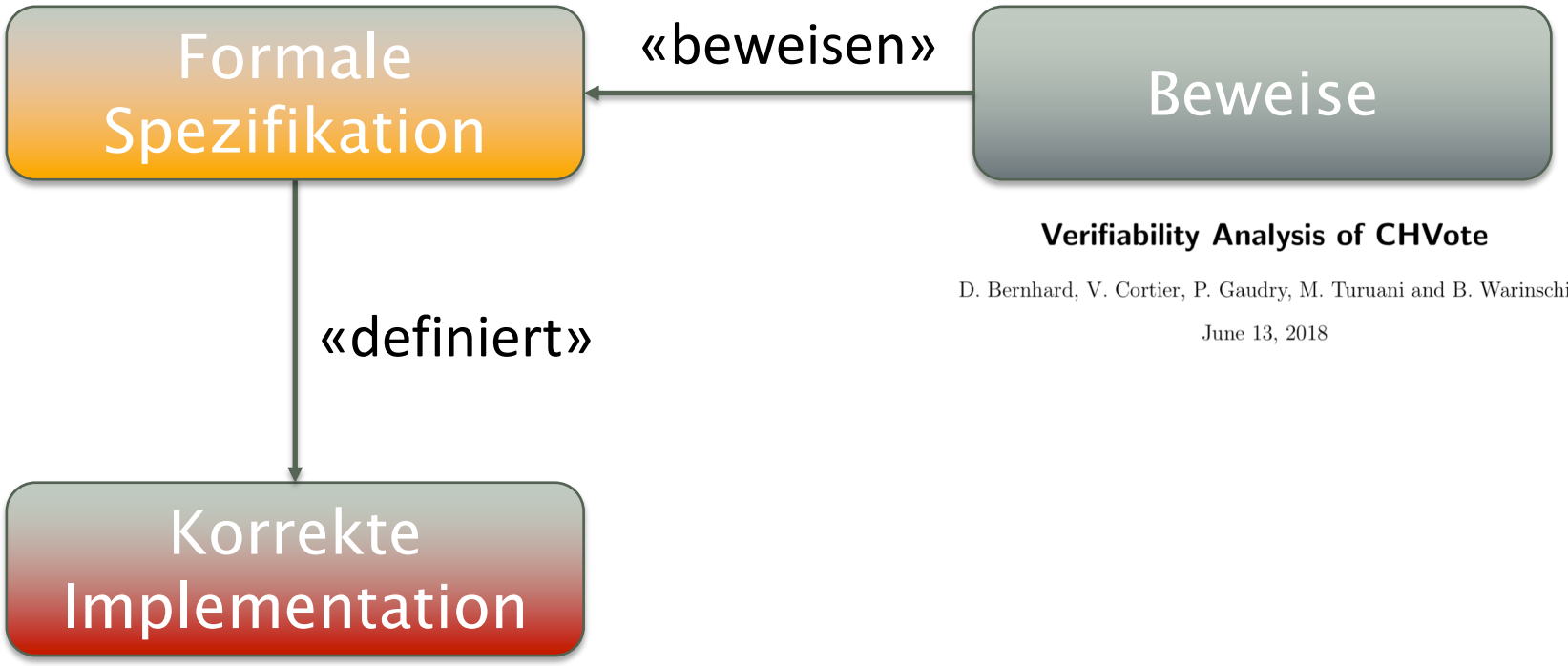
- End-zu-End-Verschlüsselung
- Verifizierbarkeit
  - **individuelle** Verifizierbarkeit
    «Wurde meine Stimme unverändert vom System erfasst?»
  - **universelle** oder vollständige (in VEleS) Verifizierbarkeit
    «Wurden alle korrekt erfassten, gültigen Stimmen (und nur solche) richtig gezählt?»
- Aufteilung von «Vertrauen» auf mehrere unabhängige Systemteile (*distribution of trust*)
  - verifizierbares Mix-Net
  - Splitten des Entschlüsselungsschlüssels

Grobstruktur Schweizer E-Voting-Systeme



Was braucht es?

https://eprint.iacr.org/2017/325



Verifiability Analysis of CHVote

D. Bernhard, V. Cortier, P. Gaudry, M. Turuani and B. Warinschi

June 13, 2018

Beispiel: Von der «Spezifikation» ...

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

Input: Voting code $X \in A_X^{\ell_X}$
Selection $\mathbf{s} = (s_1, \ldots, s_k)$, $1 \leqslant s_1 < \cdots < s_k$
Encryption key $pk \in \mathbb{G}_q \backslash \{1\}$

$x \leftarrow$ ToInteger(X) // see Alg. 4.7
$\hat{x} \leftarrow \hat{g}^x \bmod \hat{p}$
$\mathbf{q} \leftarrow$ GetSelectedPrimes$(\mathbf{s})$ // $\mathbf{q} = (q_1, \ldots, q_k)$, see Alg. 7.19
$m \leftarrow \prod_{i=1}^{k} q_i$
if $m \geqslant p$ **then**
  return $\perp$ // (k, n) is incompatible with p
$(\mathbf{a}, \mathbf{r}) \leftarrow$ GenQuery$(\mathbf{q}, pk)$ // $\mathbf{a} = (a_1, \ldots, a_k)$, $\mathbf{r} = (r_1, \ldots, r_k)$, see Alg. 7.20
$a \leftarrow \prod_{i=1}^{k} a_i \bmod p$
$r \leftarrow \sum_{i=1}^{k} r_i \bmod q$
$b \leftarrow g^r \bmod p$
$\pi \leftarrow$ GenBallotProof$(x, m, r, \hat{x}, a, b, pk)$ // $\pi = (t, s)$, see Alg. 7.21
$\alpha \leftarrow (\hat{x}, \mathbf{a}, b, \pi)$
return $(\alpha, \mathbf{r})$ // $\alpha \in \mathbb{Z}_{\hat{q}} \times \mathbb{G}_q^k \times \mathbb{G}_q \times ((\mathbb{G}_{\hat{q}} \times \mathbb{G}_q^2) \times (\mathbb{Z}_{\hat{q}} \times \mathbb{G}_q \times \mathbb{Z}_q))$, $\mathbf{r} \in \mathbb{Z}_q^k$

Algorithm 7.18: Generates a ballot based on the selection $\mathbf{s}$ and the voting code X.

... zur Implementation

```java
/**
 * Algorithm 7.18: GenBallot
 *
 * @param upper_x the voting code
 * @param bold_s  voters selection (indices)
 * @param pk      the public encryption key
 * @return the combined ballot, OT query and random elements used
 */
public BallotQueryAndRand genBallot(String upper_x, List<Integer> bold_s, EncryptionPublicKey pk) {
    BigInteger x = conversion.toInteger(upper_x, publicParameters.getUpper_a_x());
    BigInteger x_circ = modExp(g_circ, x, p_circ);
    List<BigInteger> bold_q = computeBoldQ(bold_s);
    BigInteger m = computeM(bold_q, p);
    ObliviousTransferQuery query = genQuery(bold_q, pk);
    BigInteger a = computeA(query, p);
    BigInteger r = computeR(query, q);
    BigInteger b = modExp(g, r, p);
    NonInteractiveZKP pi = genBallotProof(x, m, r, x_circ, a, b, pk);
    BallotAndQuery alpha = new BallotAndQuery(x_circ, query.getBold_a(), b, pi);

    return new BallotQueryAndRand(alpha, query.getBold_r());
}
```

Auszug (mit Kürzung) aus: CHVote Prof-of-Concept

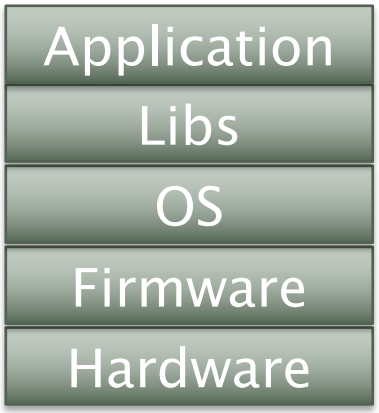
Vertrauensannahmen

Korrektheit des **Ergebnisses**, **Stimmgeheimnis** (sonst wird es gemerkt):

- Druckerei muss korrekt arbeiten
- Maximal eine Kontrollkomponente muss korrekt arbeiten

Privacy

- Kein «*side channel*» beim Gerät des Stimmenden



Zusammenfassung

Zusammenfassung

- Das Stimmgeheimnis kann durch geeignete kryptografische Verfahren garantiert werden
  - z.B. kryptografisches Mix-Net
    - mehrere unabhängige Mischstufen (Kontrollkomponenten)
    - Permutation der Stimmen
    - Wiederverschlüsselung der Stimmen
    - kryptografischer Beweis fürs korrekte Permutieren
    - kryptografische Beweise für die korrekte Wiederverschlüsselungen
- Die universelle Verifikation ist möglich, ohne das Stimmgeheimnis zu gefährden

Vielen Dank

Prof. Dr. Eric Dubuis
Berner Fachhochschule
RISIS
2501 Biel
Switzerland