Non-interactive Zero-knowledge and its applications

including e-voting

Vincenzo Iovino

University of Luxembourg

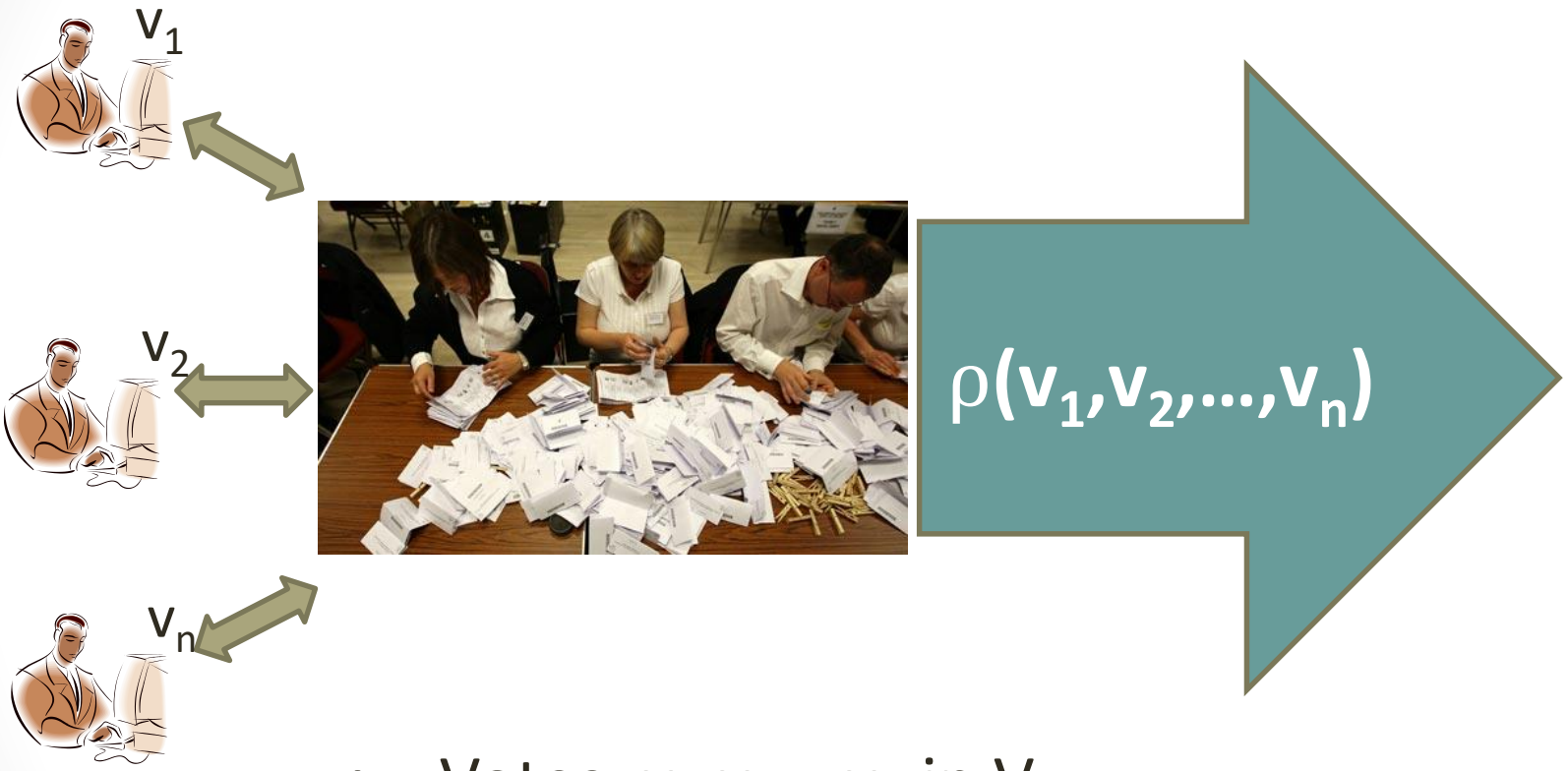
vinciovino@gmail.com

part of these slides are courtesy of B. Warinschi

ZK protocols

Nowdays, ZK became popular for its applications to cryptocurrencies but since its invention ZK have had a tremendous impact in the design of **verifiable** e-voting schemes

Voting scheme



- Votes: $v_1,v_2,\ldots v_n$ in V
- Result function: $\rho : V^* \rightarrow$ Results
- E.g. (referendum) V={0,1},
  $\rho(v_1,v_2,\ldots,v_n)= v_1+v_2+\ldots+v_n$

E-voting: wished properties

- **Eligibility**: only legitimate voters vote; each voter votes once
- **Verifiability**: individual, universal
- **Privacy**: no information about the individual votes is revealed
- **Coercion-resistance** : a voter cannot interact with a coercer to prove that she voted in a certain way

Non-verifiable e-voting

$P_1: v_1$

$C_1 \leftarrow ENC_{PK}(v_1)$

$P_2: v_2$

$C_2 \leftarrow ENC_{PK}(v_2)$

$P_n: v_n$

$C_n \leftarrow ENC_{PK}(v_n)$

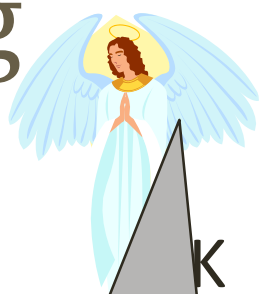
PK

BB

C_1

C_2

C_n



K

Use SK to obtain $v_1,\ldots v_n$. Compute and return $\rho(v_1,v_2,\ldots,v_n)$

Homomorphic Tallying

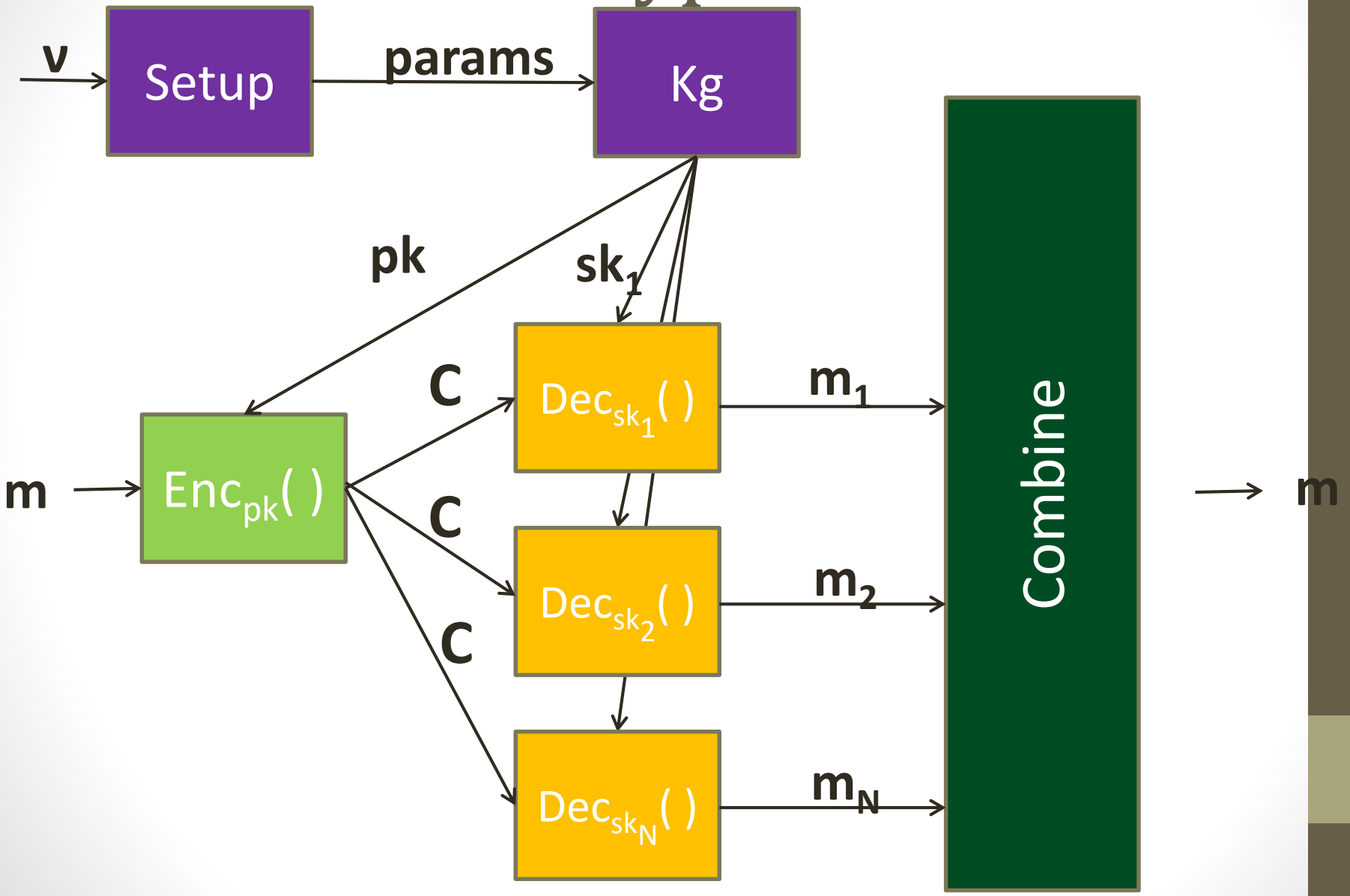
- Let π=(KG,ENC,DEC) be a homomorphic encryption scheme. Enc2Vote(π) is:
- **Setup(v)**: KG generates (SK,PK,[])
- **Vote(PK,v)**: b $\leftarrow$ $ENC_{PK}(v)$
- **Process Ballot([BB],b)**: [BB] $\leftarrow$ [BB,b]
- **Tallying([BB],x)**: where [BB] = $[b_1,b_2,\ldots,b_n]$

  $b = b_1 \cdot b_2 \cdot \ldots \cdot b_n$

  result $\leftarrow DEC_{SK}(x,b)$

  output **result**

Threshold encryption



Mixnets

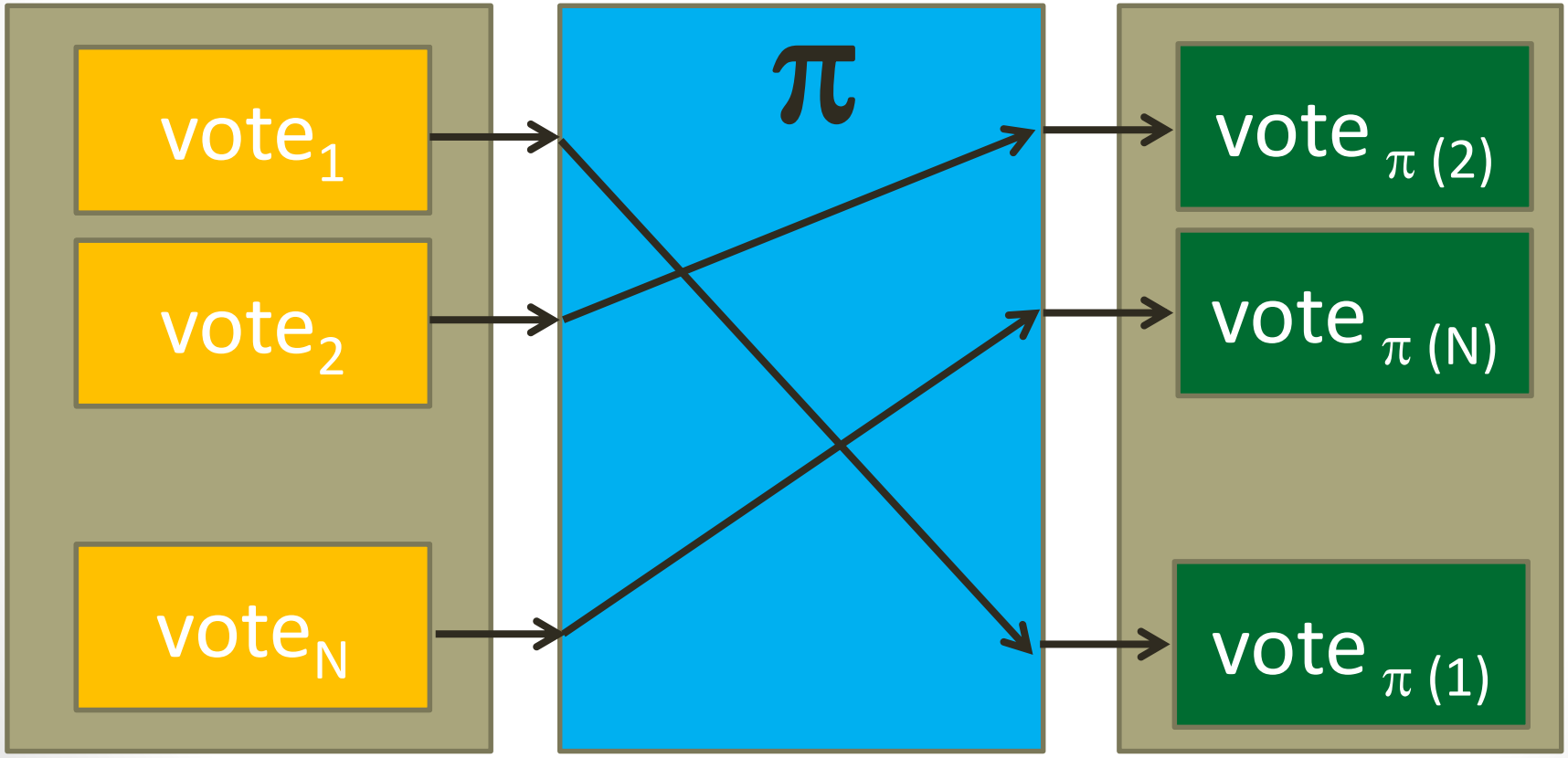
- Homomorphic tallying great, but not for complex functions
  - Instead of homomorphically computing $Enc_{pk}(f(v_1,v_2,\ldots,v_n))$ simply decrypt all votes

Rerandomizable encryption

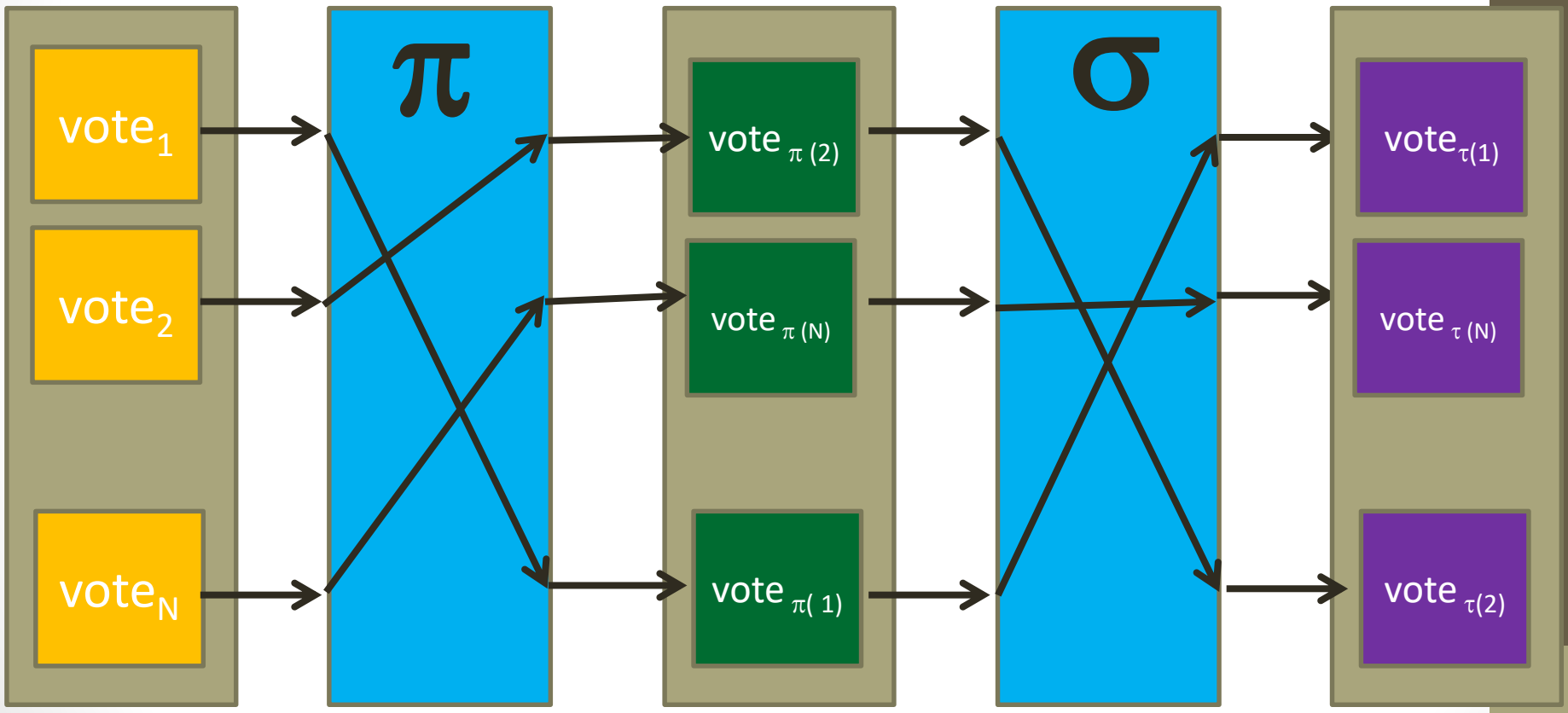
$$\mathbf{Enc_{pk}(m;r) \bullet Enc_{pk}(0;s) = Enc_{pk}(m;r+s)}$$

$$(g^r, g^m X^r) \cdot (g^s, g^0 X^s) = (g^{r+s}, g^m X^{r+s})$$

Mixnet

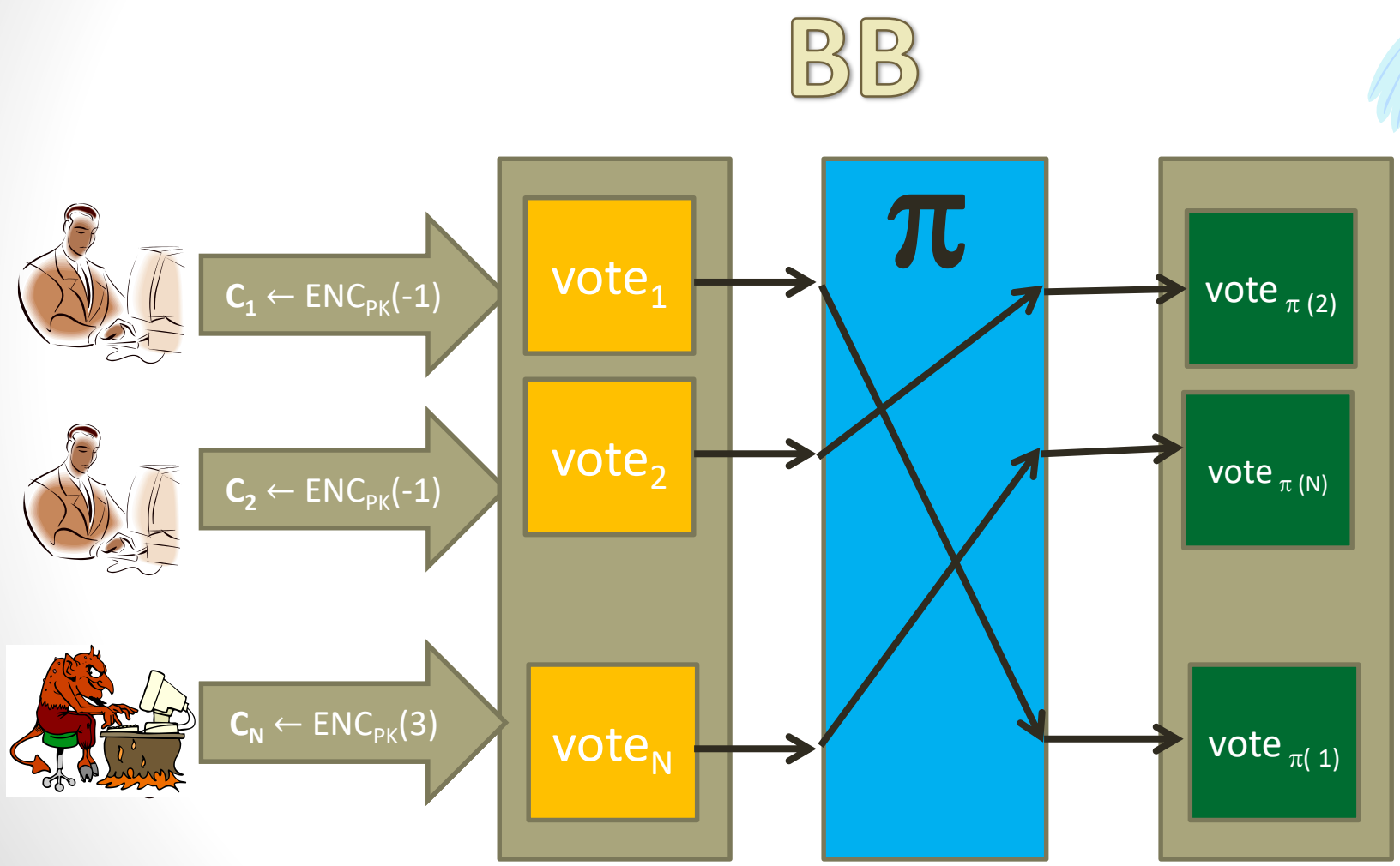


Mixnet

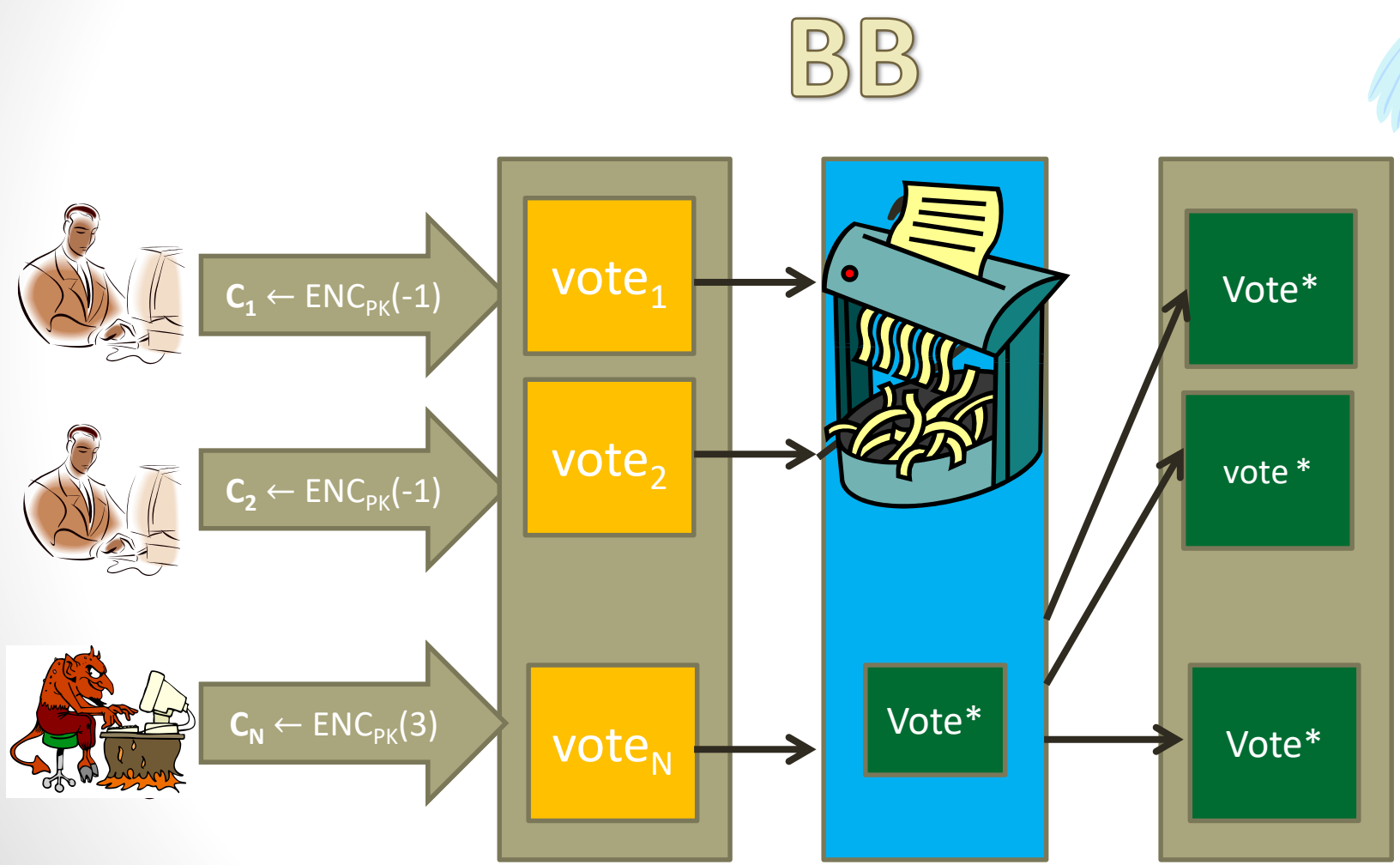


$\tau = \pi ; \sigma$

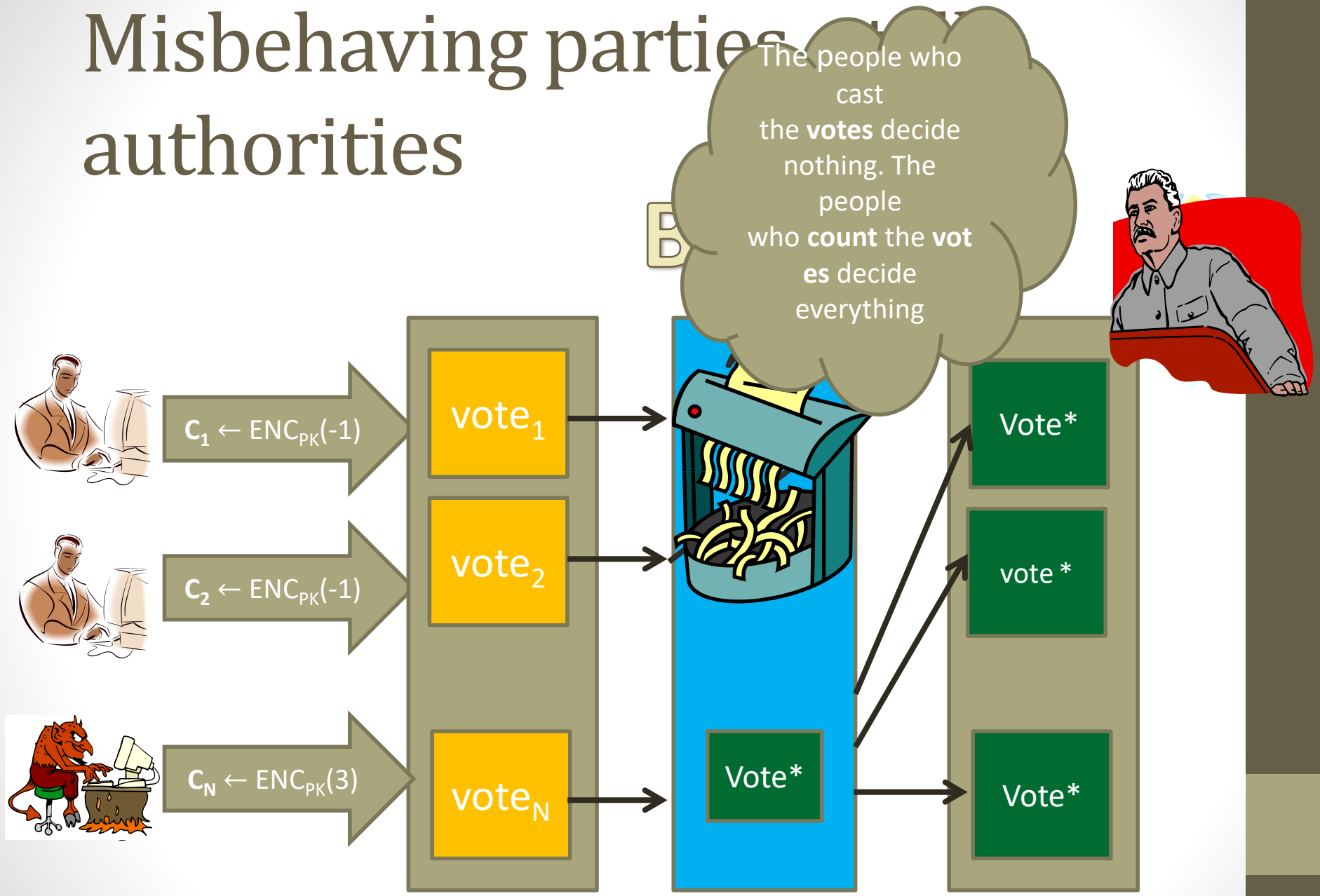
Misbehaving parties - voters



Misbehaving parties - mixers



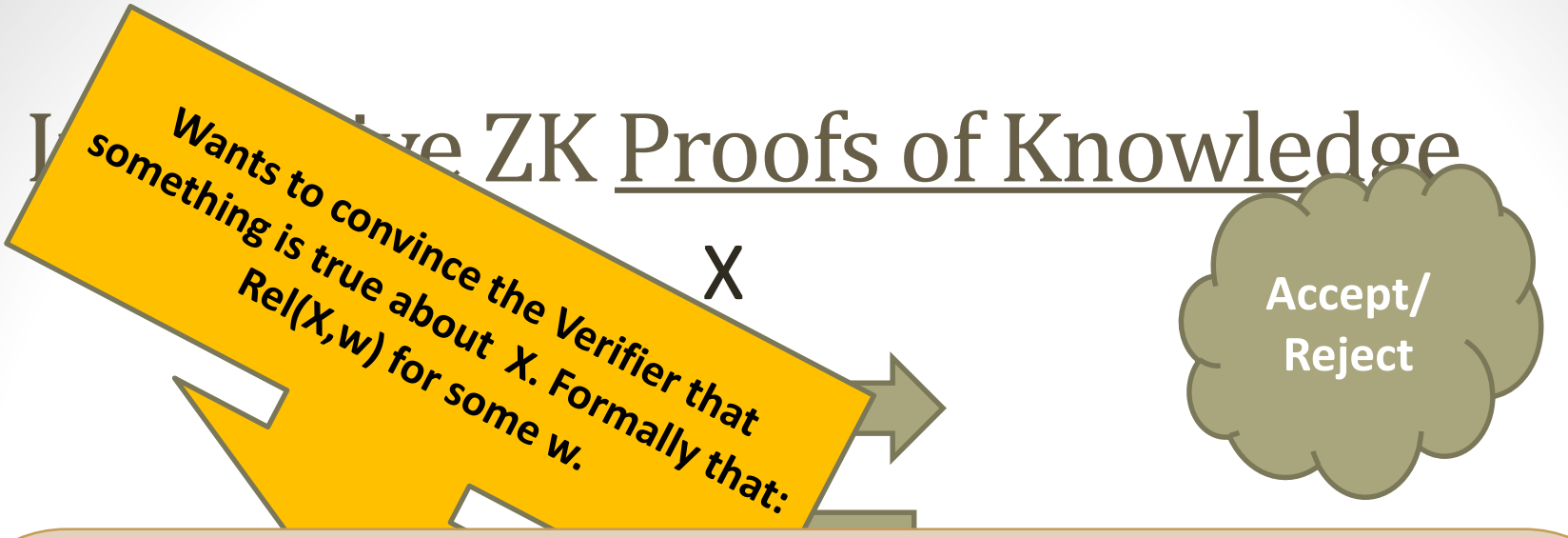
Misbehaving parties authorities



Misbehaving parties

- **Voters**: non-well formated votes; problematic for homomorphic tallying
- **Mixservers**: may completely replace the encrypted votes
- **Tallying authorities** : may lie about the decryption results

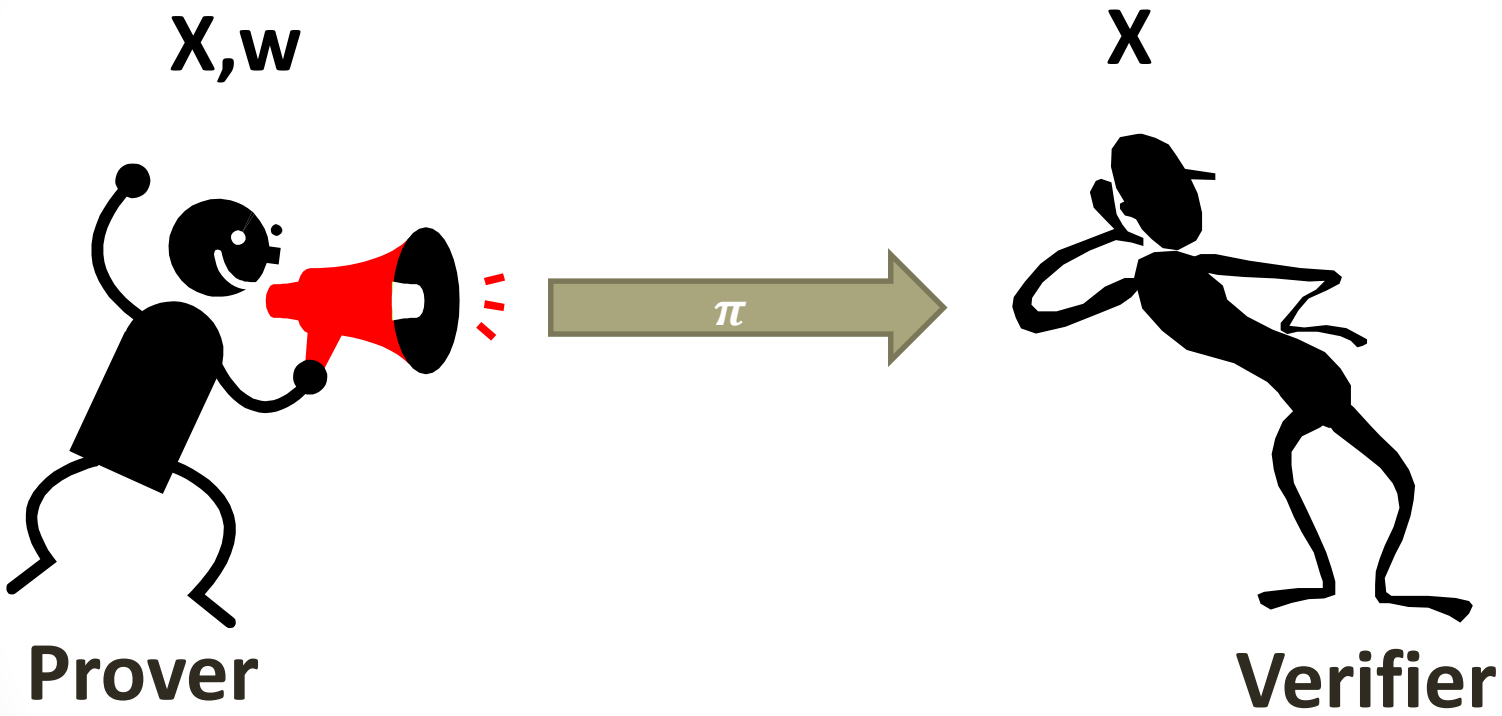
Interactive ZK Proofs of Knowledge



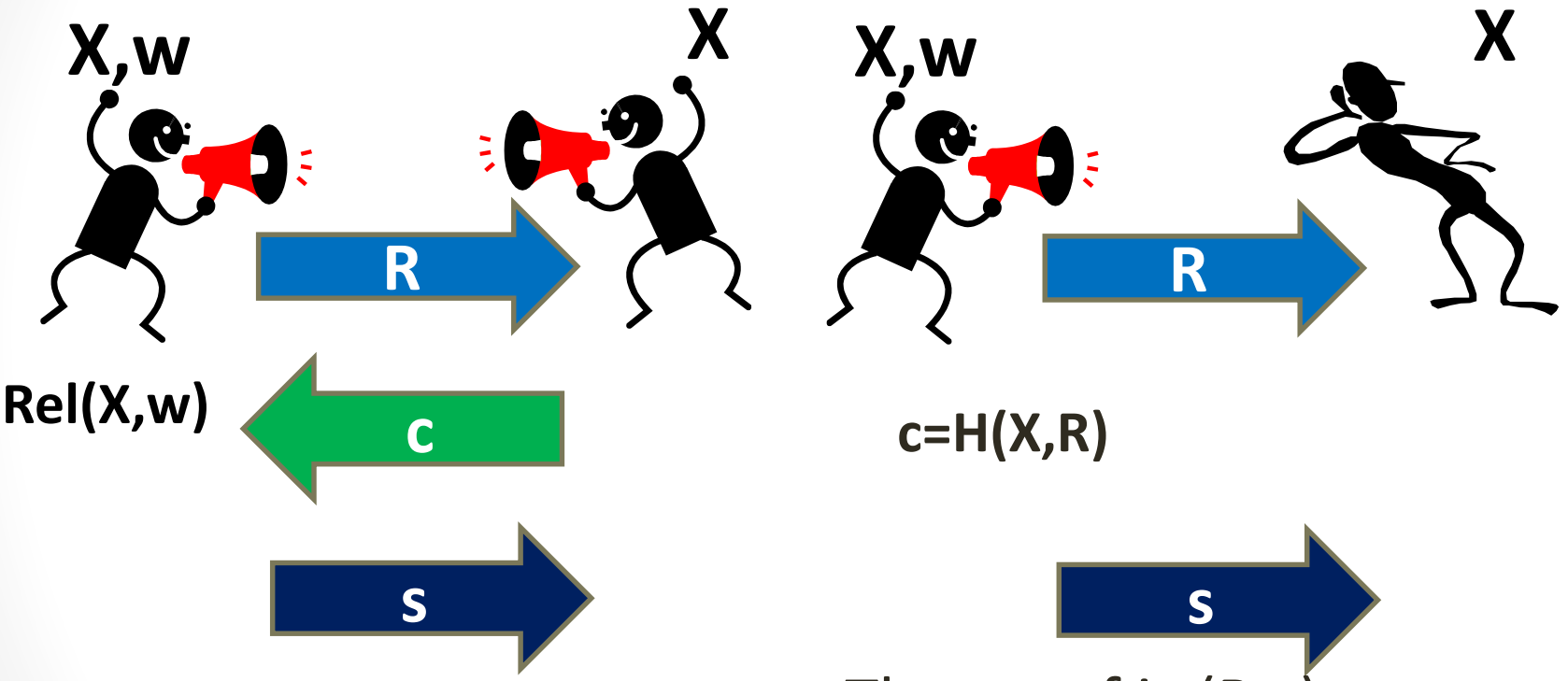
Examples:

- **The ciphertext encrypts “Obama” OR “Trump”**
- **The ciphertext decrypts to value 11245 (11245 votes for YES in a referendum).**
- **The ciphertexts (c,d) are a shuffle of (a,b)**

Universal Verifiability via non-interactive proofs



The Fiat-Shamir transform



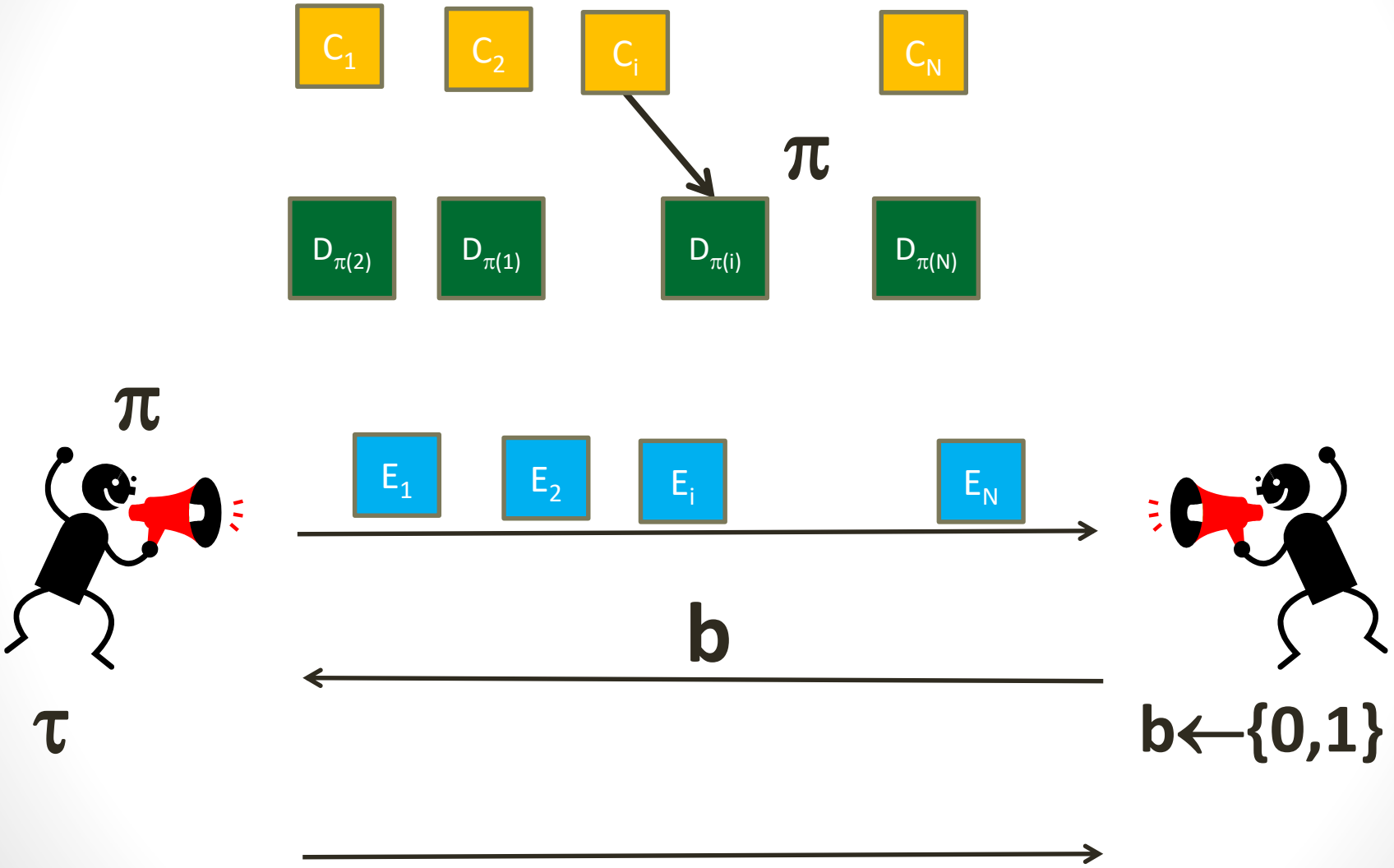
To verify: check (R,c,s) as before.

The proof is (R,s).
To verify: compute c=H(R,s). Check (R,c,s) as before

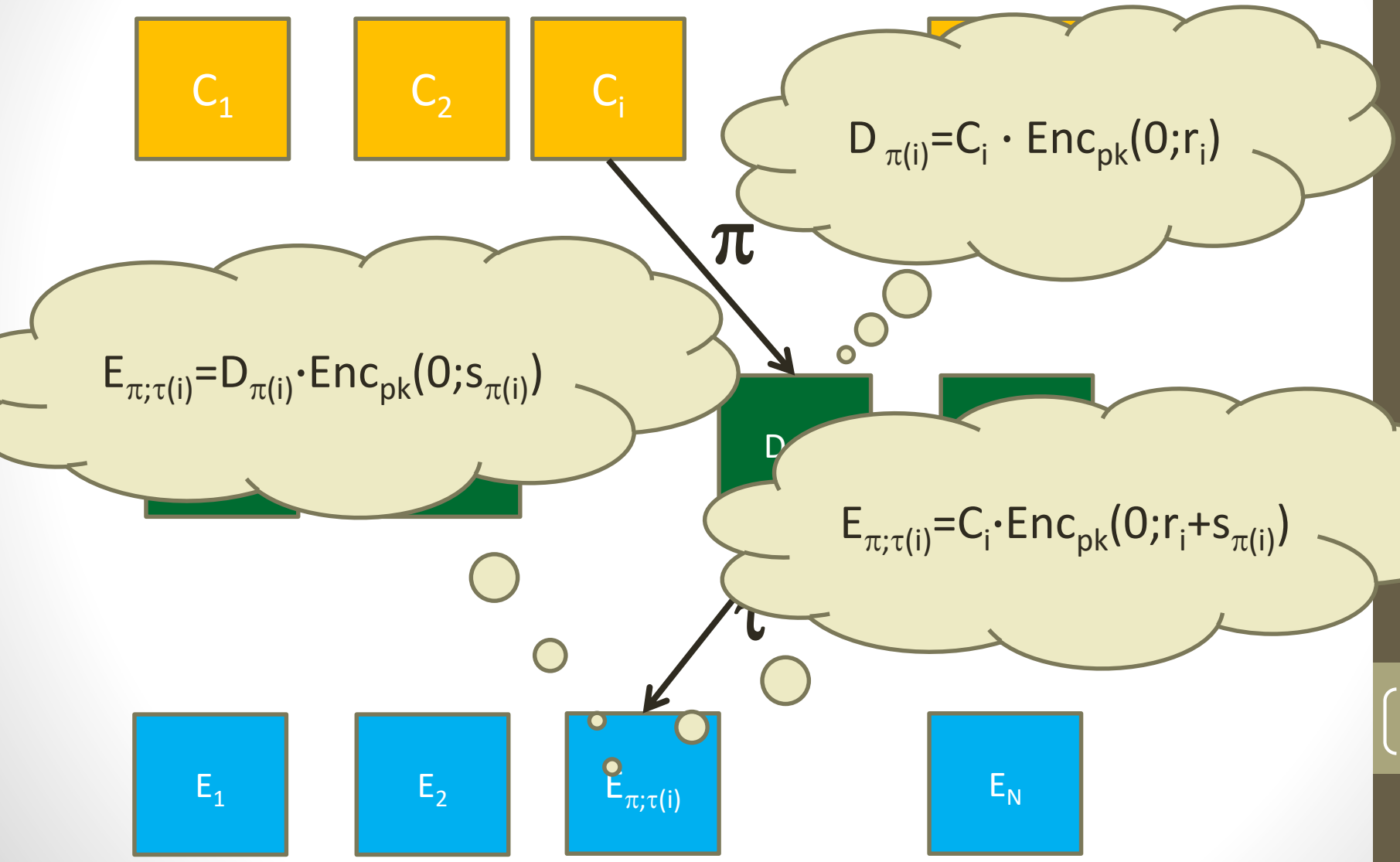
Applications of NIZKs

- Proofs of correct decryption for tallying based on threshold decryption (dishonest tallies)
- Verifiable Mixnets/Shuffles (dishonest mixers)
- Achieve **Universal Verifiability**: not only who participated in the election but every party should be convinced about the election's result

Verifiable shuffle [KS95]



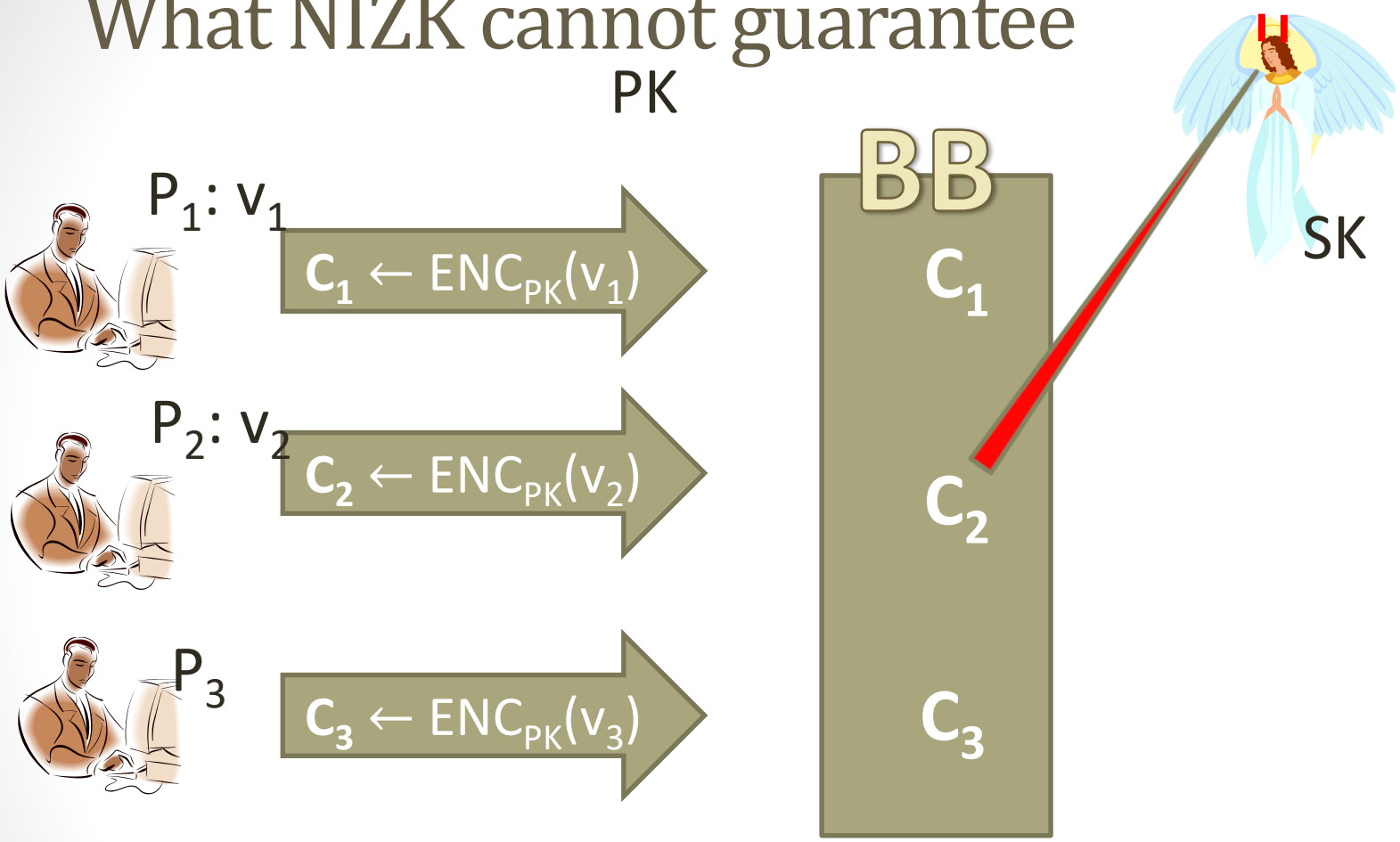
Verifiable shuffle [KS95]



E-voting and cryptocurrencies

- These apparently different protocols rely on similar building blocks and properties
- **Zcash** preserves privacy of the transactions by means of ZK
- Range proofs are used in e-voting to prove that a voter casts a valid ballot found out new applications in **Confidential Transactions**
- The development of **SNARKs** fostered by cryptocurrencies led to the design of new super fast verifiable **shuffles**
- **Universal verifiability** fundamental also for cryptocurrencies (a SNARK proof proof of a shielded transaction should convince everybody even in the future)

What NIZK cannot guarantee



- The authority can DELETE from the BB (signatures do not help).
- **Blockchains** may help in distributing the trust of the BB

What NIZK cannot guarantee

- Both in e-voting and cryptocurrencies, the soundness is significantly more important than ZK:

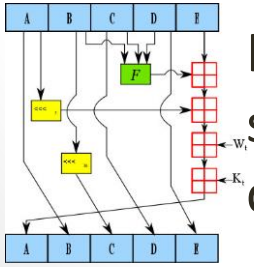
If an authority can break the soundness of the proof can completely **subvert** the result of the election

If an adversary can break the soundness of a SNARK, can create **infinite coins** in Zcash

  - Unluckily, the soundness is **not** unconditional

ZK vs Soundness

- ZK and soundness are two **conflicting** properties
- Perfect ZK cannot be achieved unless weaking the soundness to be computational
- NIZKs cannot be achieved in the plain model.
- Need for trusted parameters or the RO (soundness is **NOT** unconditional)
- Note: the RO is still a **trust assumption:**



If a trapdoor is discovered in the hash function, the security of the e-voting or cryptocurrency protocol is completely comprised

Comparing trust assumptions

- In the **CRS** model (Zcash, several SNARKs), a so called common reference string (CRS) has to be produced. If the CRS is correctly generated, it is difficult to prove false theorems (subvert election or forge coins).
- The generator of the CRS could generate it such that he can prove false theorems.
- In the **Random Oracle** (RO) model one assumes that an hash function behaves as a ‘’good’’ random function.
- If this is not the case, or the function has a trapdoor, there may **exist** proofs for false theorems.
- What is preferable? After the Zcash’s flaw, the community is pushing towards adoption of RO-based proofs (STARKs, Bulletproofs, etc.)

Future directions

- Apply recent progresses in Bulletproofs to e-voting, e..g, verifiable shuffles
- Improve the soundness of Bulletproofs et similia
- Alternative models of ZK with better soundness guarantees

The end

Contact: vinciovino@gmail.com

part of these slides are courtesy of B. Warinschi