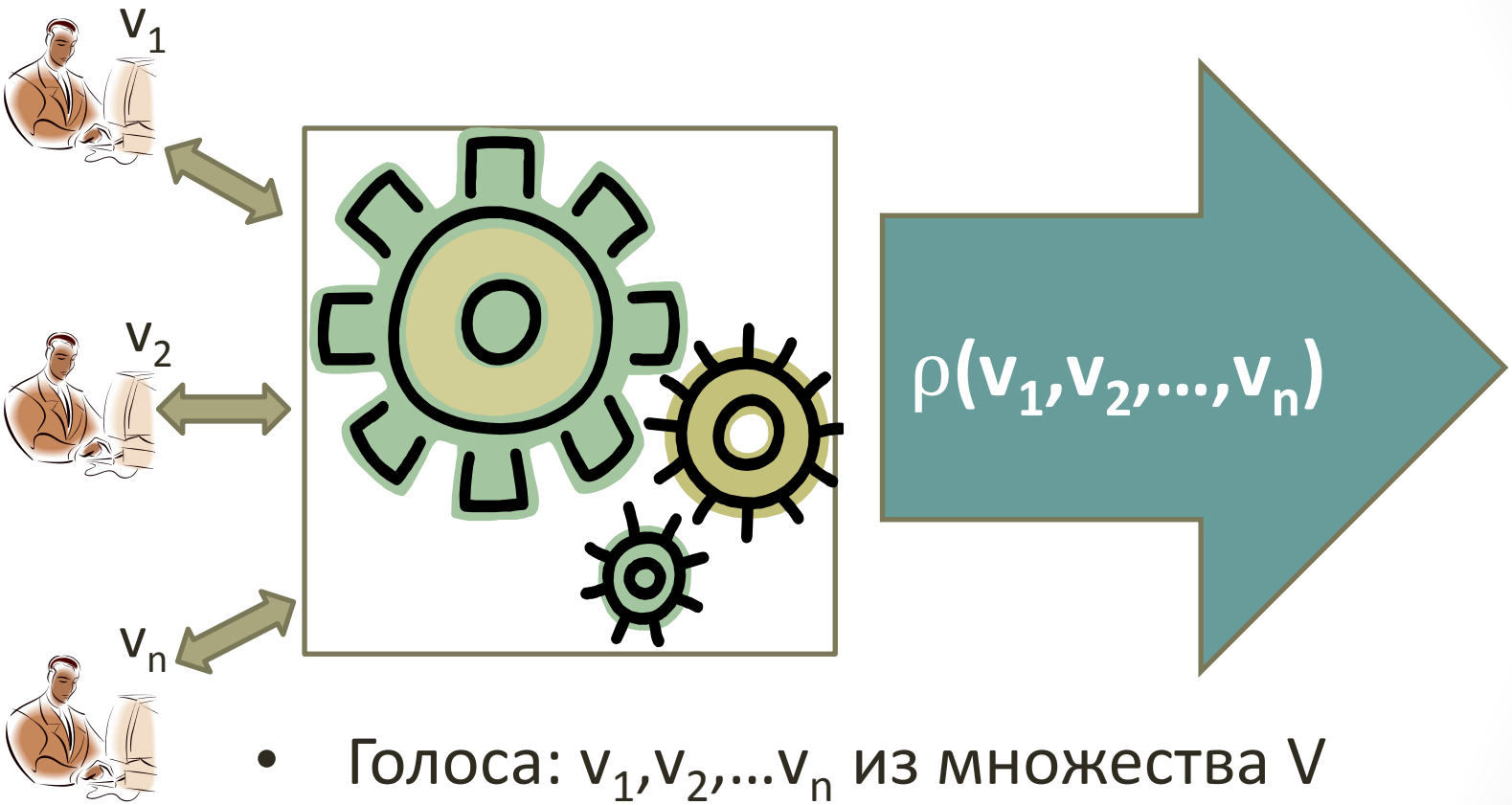
Электронное голосование и цифровые деньги

Схема голосования



- Голоса: $v_1,v_2,\ldots v_n$ из множества V
- Результирующая функция: $\rho : V^* \rightarrow$ Results
- Например: V={0,1},
- $\rho(v_1,v_2,\ldots,v_n)= v_1+v_2+\ldots+v_n$

Сложные выборы

- 2 кандидата; Принятие решению по максимуму голосов
- N кандидатов:
  - **Ограниченное голосование**: каждый голосующий может проголосовать за t<N кандидатов
  - **Утверждающие выборы**: можно голосовать за любое количество кандидатов
  - **Делимое голосование**: t голосов делятся между всеми кандидатами
  - **Голосование Borda** : голоса определяют ранг кандидата

Требования к схеме

- **Приемлемость**: голосовать могут только имеющие права; каждый может голосовать только 1 раз
- **Честность**: голосование не выявляет ранних результатов
- **Проверяемость**: индивидуальная, универсальная
- **Конфиденциальность**: информация об отдельных голосах отсутствует
- **Receipt-freeness:** невозможность продать голос
- **Coercion-resistance** : свобода от принуждения

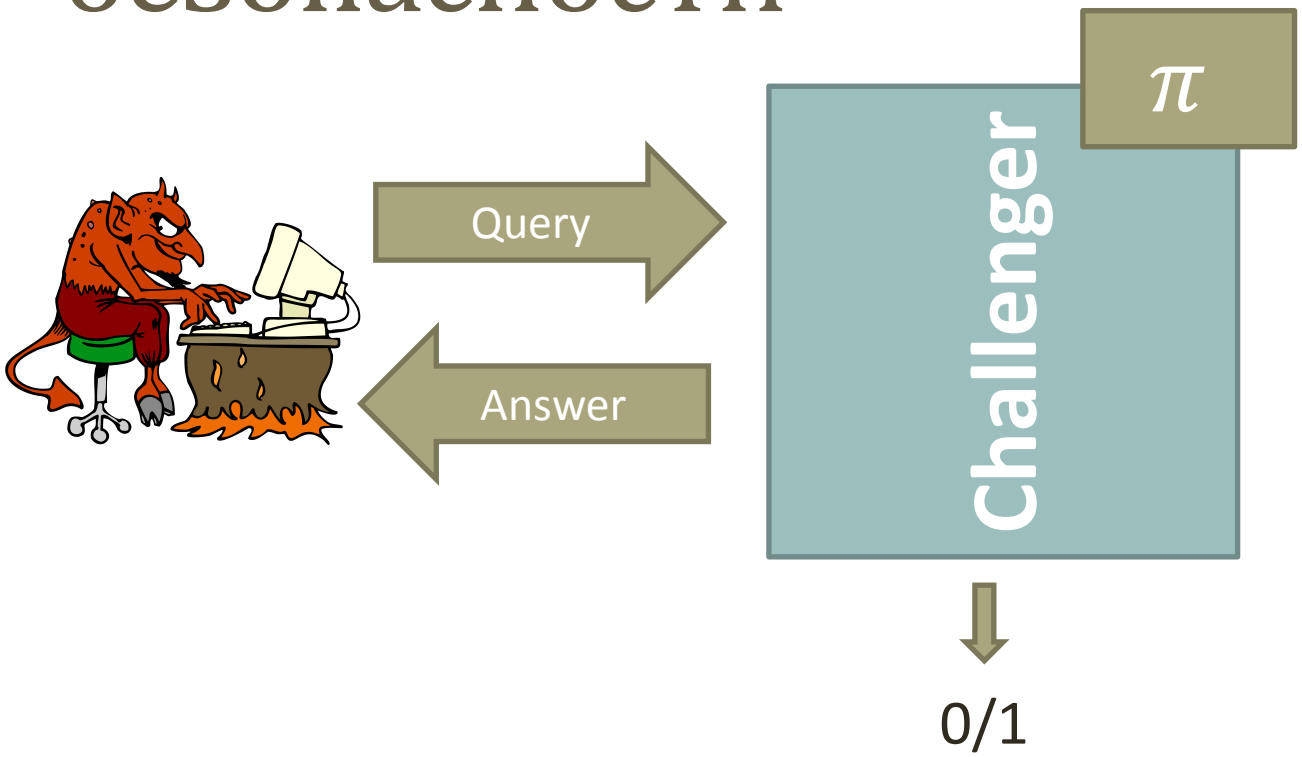
Конфиденциальность

- **Криптографические примитивы**
  - Схемы обязательств, слепые подписи, асимметричная криптография, разделение секрета
- **Криптографические техники**
  - Гомоморфизмы, рандомизация, пороговая криптография
- **Модели безопасности:**
  - для примитивов, и безопасности голосов и бюллетеней
- **Схемы голосования:**
  - FOO, Minivoting scheme

Проверяемость

- **Криптографические доказательства**
  - Схемы с нулевым разглашением
  - Приложения нулевого разглашения
- **Схема голосования в Интернет Helios**

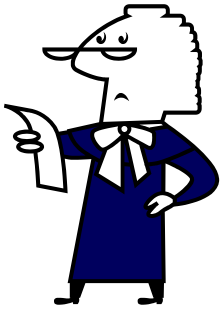
Игровые модели безопасности



Безопасность: π будем называть безопасной, если для любого атакующего вероятность, что челенжер вернет единицу близка к константе (обычно 0, или ½)

Fujisaki Okamoto Ohta [FOO92]

Голосующие



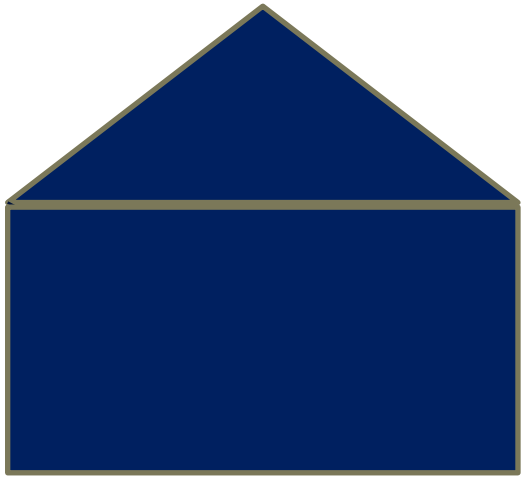
ЦУР

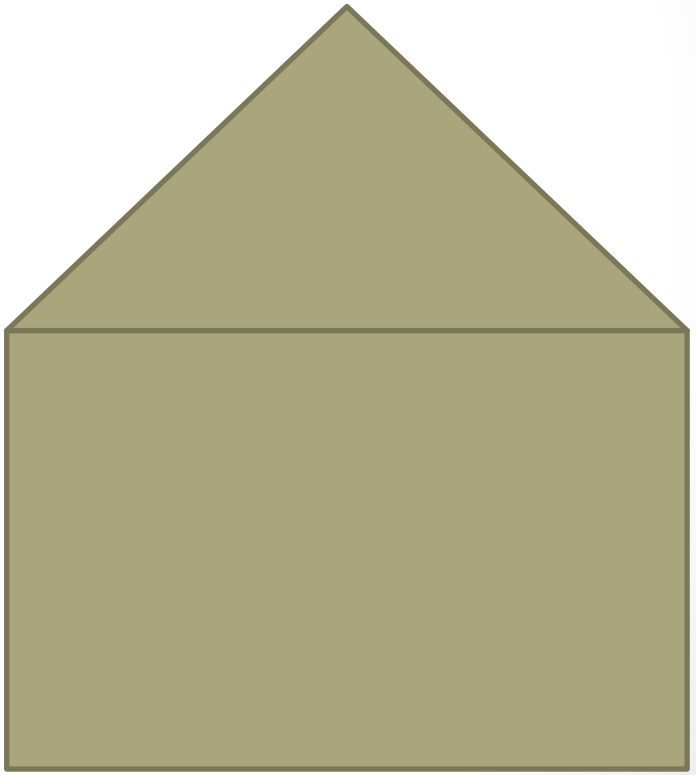
1. Фаза регистрации
2. Фаза голосования
3. Фаза подсчета

ЦИК

FOO - Регистрация

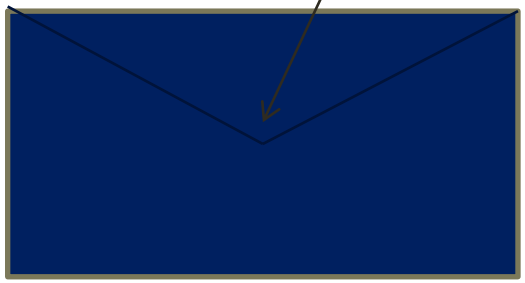
Мой голос

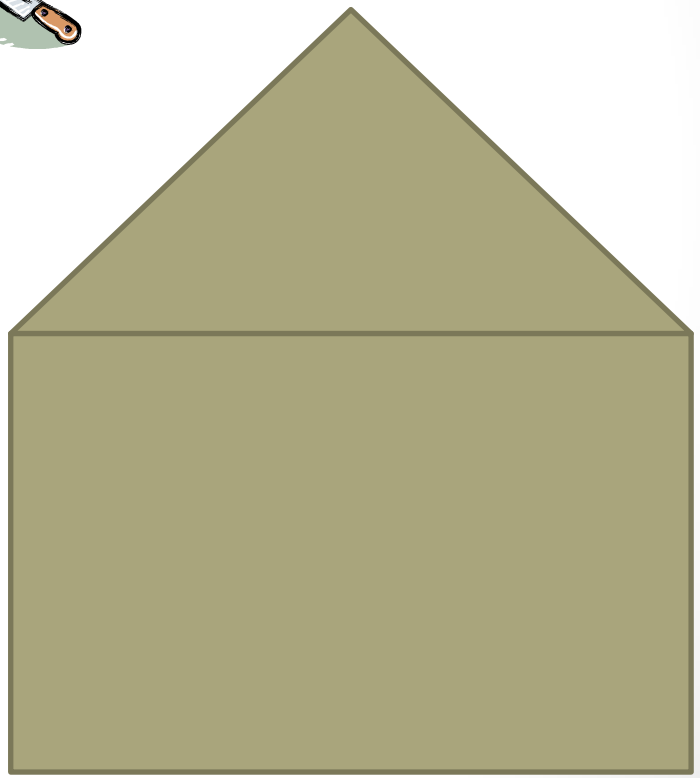




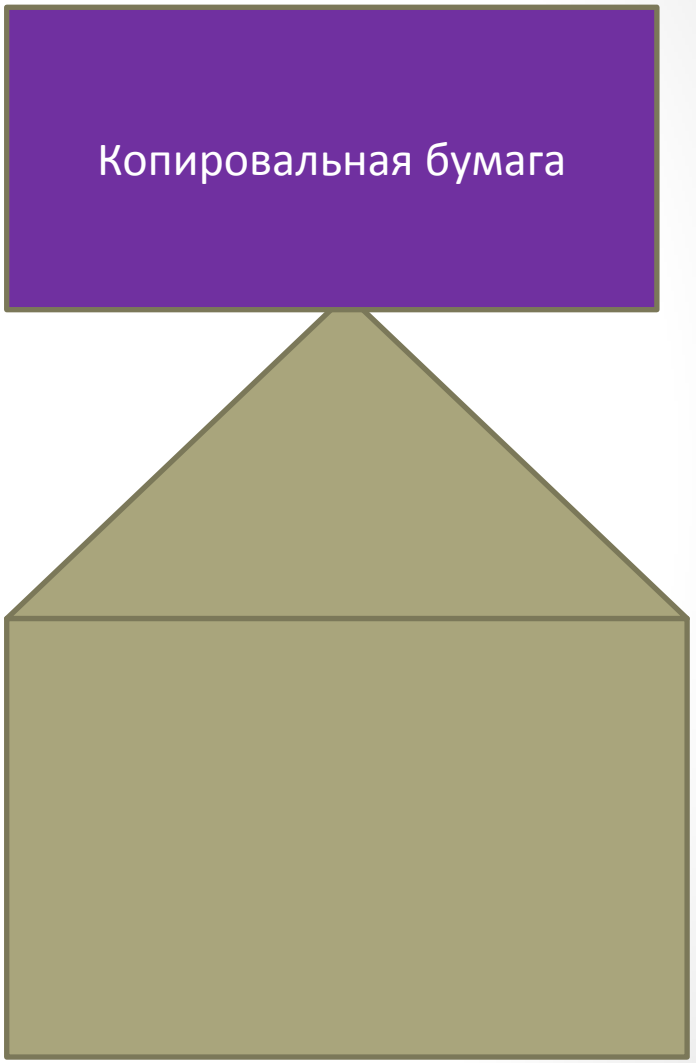
FOO - Регистрация

Специальный конверт. Его можно вскрыть только с помощью

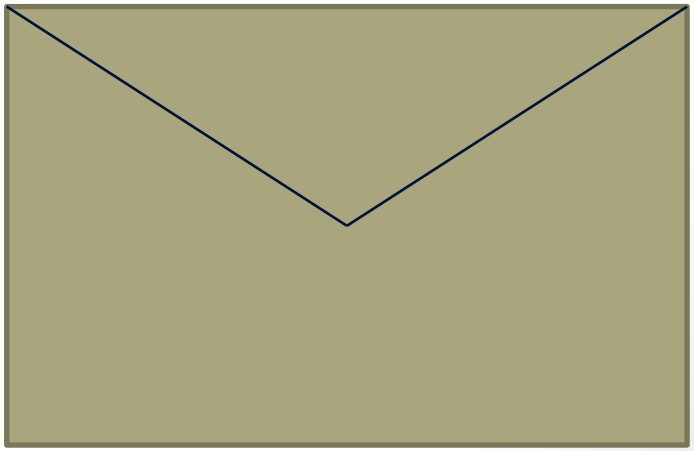




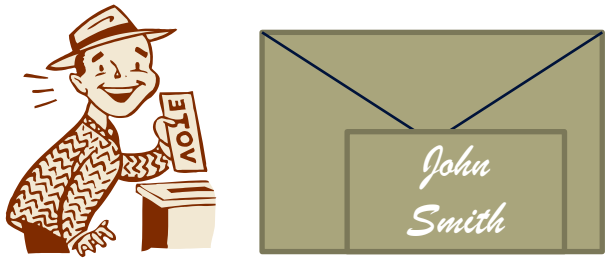
FOO - Регистрация



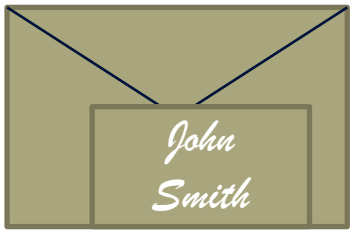
FOO - Регистрация



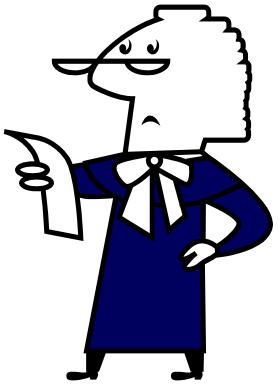
FOO - Регистрация



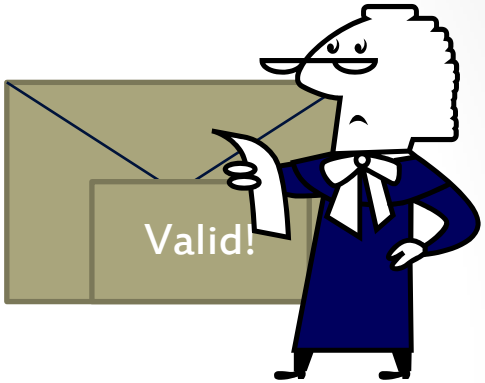
FOO - Регистрация



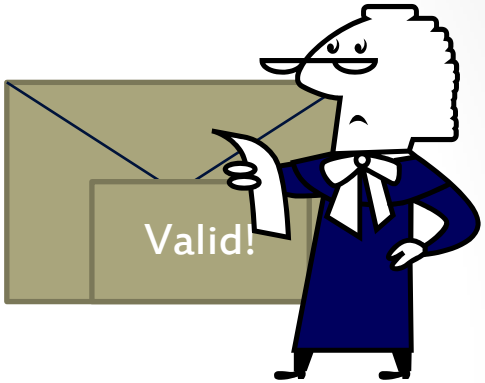
John Smith : зарегистрированный голосующий, которые еще не голосовал



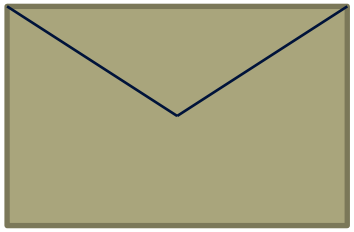
FOO - Регистрация



FOO - Регистрация



FOO - Регистрация



FOO – Фаза голосования

Valid!

Valid!

Valid!

Valid!

FOO – Фаза голосования

Valid!

Valid!

Valid!

Valid!

Анонимный канал

FOO – Фаза подсчета голосов

Анонимный канал

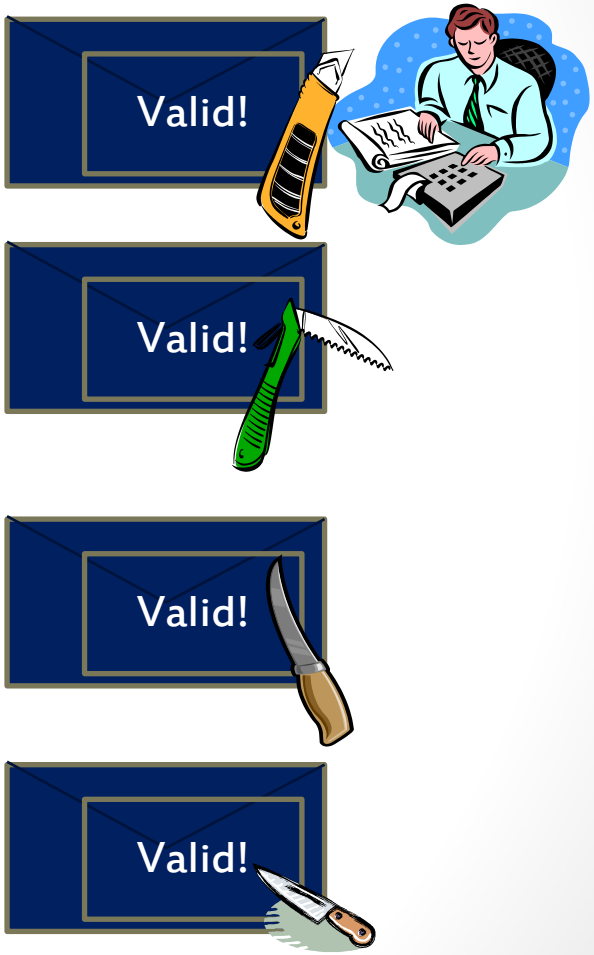
FOO – Фаза подсчета голосов

Анонимный канал

FOO – Фаза подсчета гол

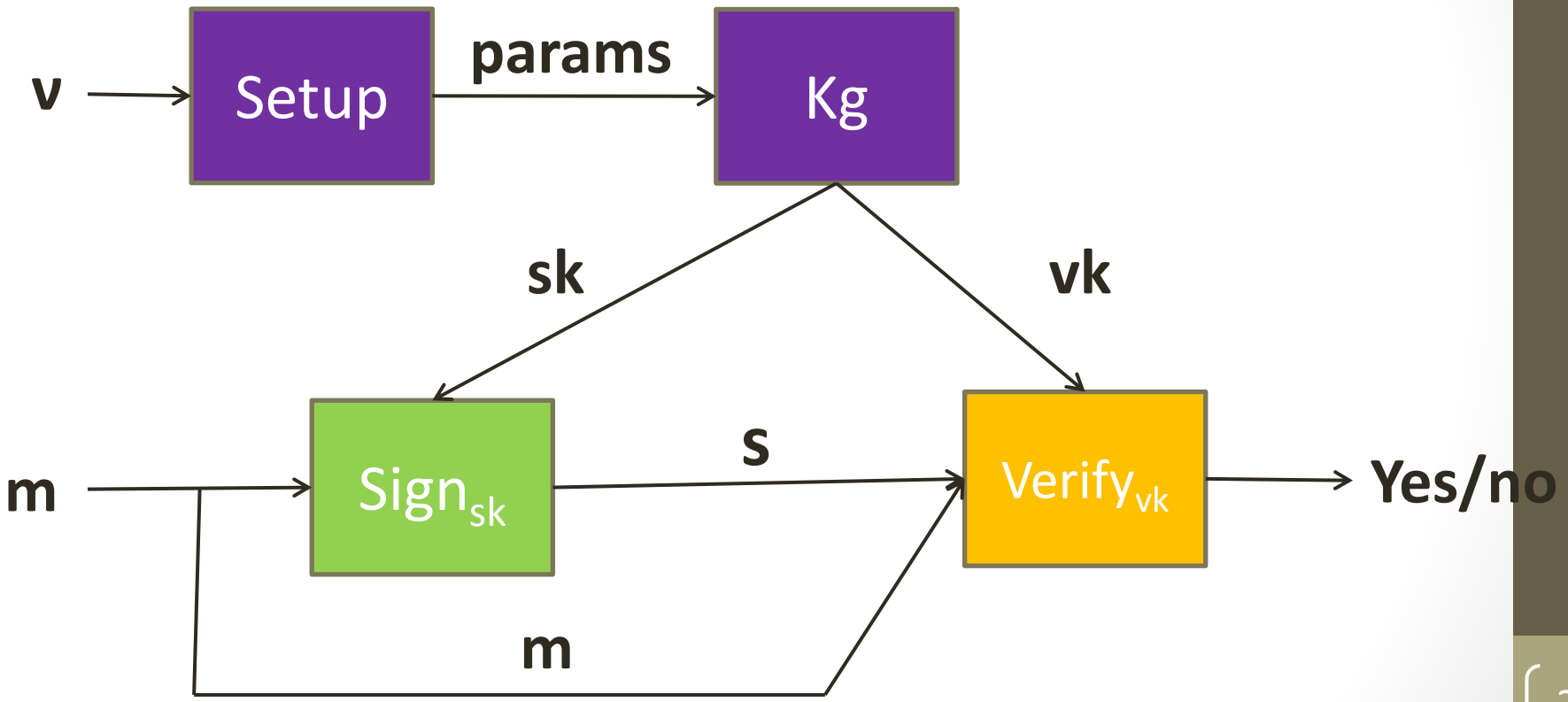
Объявляется победитель

Анонимный канал



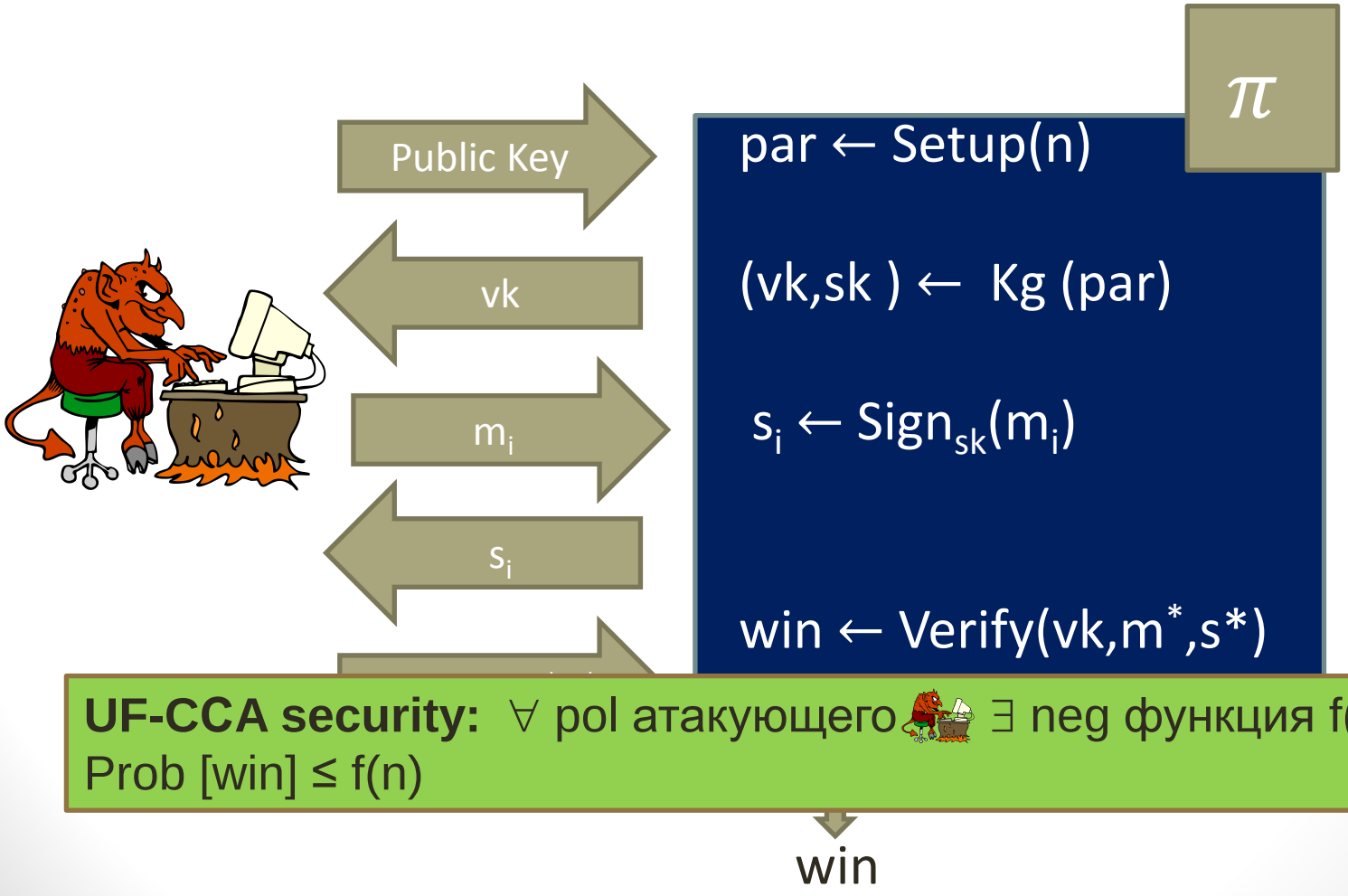
КРИПТОГРАФИЧЕСКАЯ РЕАЛИЗАЦИЯ

Цифровая подпись



Неподделываемость при CCA (UF-CCA)

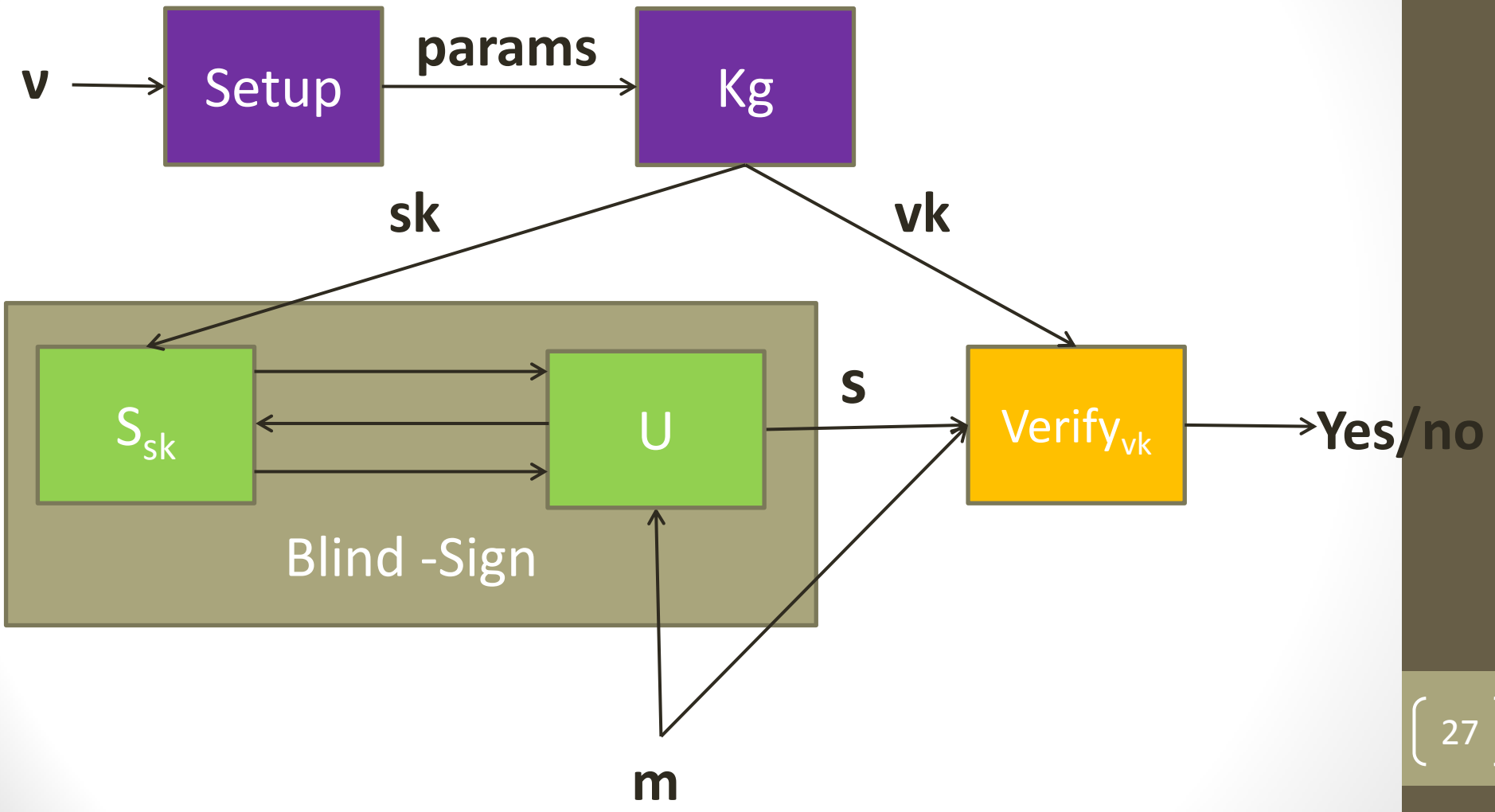
Определение стойкости π=(Setup,Kg,Sign,Verify)



Хэш функция с полной областью определения

- **Определение:**
  - **Keygen(v)**: генерация модуля RSA N=PQ, пары d и e: ed=1 mod Φ(N). Выбрать хорошую хэш-функцию H на множестве Z_N^*. Ключи: **vk**=(H,N,e) и **sk**=(H,N,d).
  - **Sign((H,N,d),m)**: подпись $H(m)^d$ mod N
  - **Verify((N,e),m,s)**:проверка s^e= H(m) mod N
- **Безопасность:** UF-CCA стойкая в модели случайного оракула и допущении RSA

Слепая цифровая подпись



Слепая цифровая подпись

- **Определение:**
  - **Keygen(v)**: генерация пары ключей (**sk**,**vk**)
  - **Blind-Sign**: протокол между пользователем U(**m**,**vk**) и подписывающим S(**sk**); пользователь получает подпись **s** для **m**
  - **Verify(vk,m,s)**: стандартный алгоритм проверки: да/нет

Слепая цифровая подпись

- **Безопасность**
  - **Слепота:** нечестный подписывающий не получает никакой информации о сообщении, которое он подписал
  - **Неподделываемость**:...

Слепая цифровая подпись Chaum'a

- **Key generation()**: построить модуль RSA N=PQ, и пару d и e такие что

$$ed=1 \bmod \Phi(N).$$

Ключи: **vk**=(N,e) и **sk**=(N,d)

- **Blind-sign:**

User (m,(N,e))

$$b = H(m)r^e \bmod N$$

$$t = b^d = (H(m)r^e)^d \bmod N$$

Signer (d,N)

$$\gcd(r, N) = 1$$

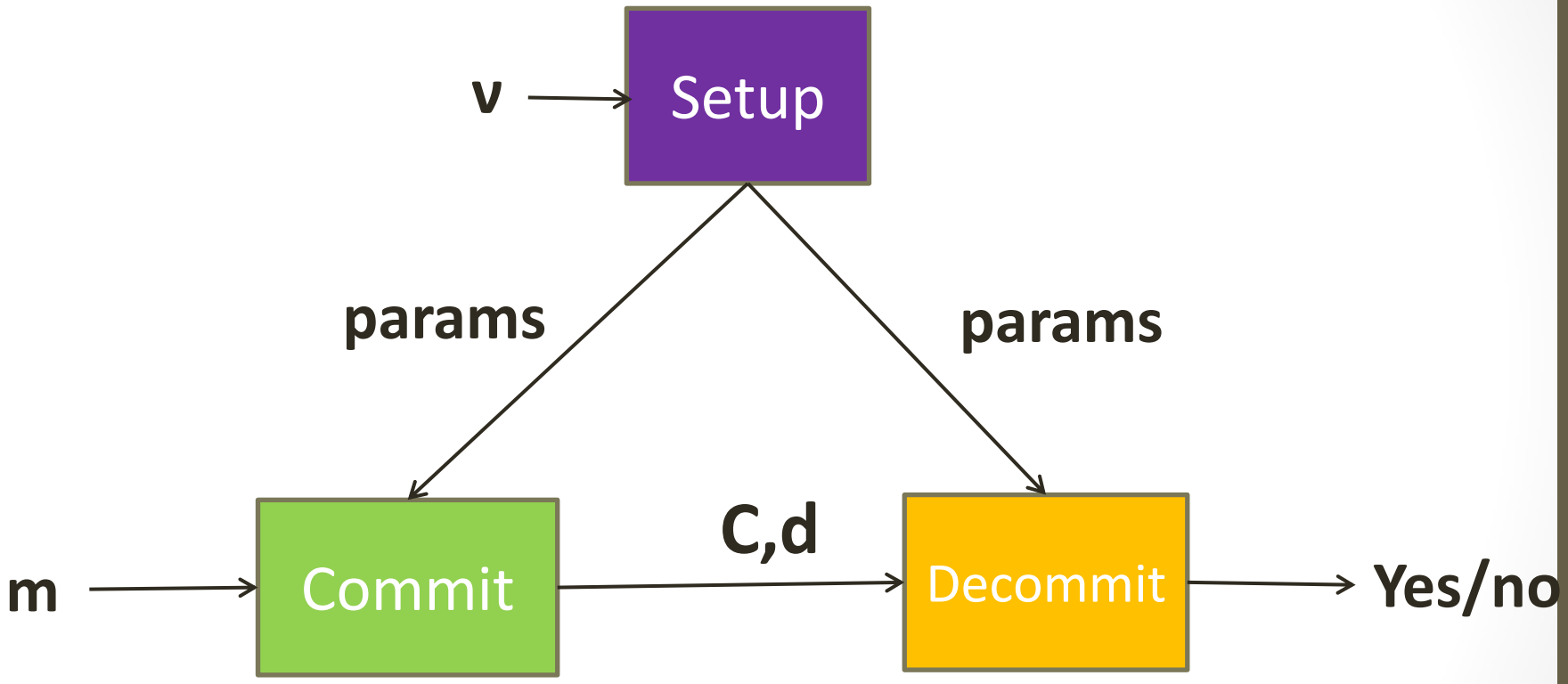
$$s = t/r = H(m)^d \bmod n$$

Схемы обязательств

- Временно скрыть значение, но при этом гарантировать, что оно не может быть изменено позже

- 1 этап: **Обязательство**
  - Отправитель «фиксирует» сообщение в электронном конверте и отправляет его получателю
- 2 этап: **Подтверждение**
  - Отправитель доказывает получателю, что в конверте находится конкретное сообщение

Схемы обязательств



Схемы обязательств

- **Определение:**
  - **Setup():** Выбирает параметры схемы
  - **Commit(x;r):** возвращает(C,d):
    - C обязательство для x
    - d информация для подтверждения
  - **Decommit(C,x,d):** возвращает да/нет
- **Требования**: Если (C,d) результат шага Commit(x;r) , тогда Decomit(C,x,d) должен вернуть да

Безопасность схемы обязательств

- **Скрытность**
  - Схема не разглашает никакой информации о передаваемом значении
    - Если в схеме доказательства получатель полиномиально ограничен, то сокрытие вычислительно стойкое; если в схеме доказательства получатель вычислительно не ограничен, то сокрытие совершенно стойкое
- **Привязка**
  - Существует не более одного сообщения, которое нечестный отравитель сможет подтвердить:
    - Совершенная и вычислительная стойкость

Давайте разберемся

- Может ли схема обязательств быть одновременно совершенно стойкой до уровню скрытности и связанности?

- Пусть G циклическая группа и g ее генератор. Рассмотрим схему (**Commit**, **Decommit**) для элементов из мн-ва {1,2,…,|G|}:
  - **Commit**(x) возвращает $C=g^x$ и d=x
  - **Decommit**(C,d) равен 1 , если $g^d=C$ и 0 иначе

- Оценить стойкость такой схемы

Схема обязательств Pedersen'a

- **Setup**: Построить циклическую группу G с простым порядком, и генератором g. Пусть
  - $h=g^a$ для случайного секрета из [|G|]
  - G,g,h публичные параметры(a хранится в секрете)

- **Commit(x;r):** чтобы зафиксировать $x \in$ [|G|], выберем случайный $r \in$ [|G|]. Обязательство для x равно $C=g^x h^r$ ($C=g^x(g^a)^r=g^{x+ar}$)

- **Decommit(C,x,r):** проверка $C=g^x h^r$

Стойкость схемы

- **Совершенное скрытие**
  - Для данного обязательства c, любое значение x равновероятно может быть скрыто в c
    - Даны x, r и любой x’, существует уникальный r’ такой что $g^x h^r = g^{x'} h^{r'}$ $r' = (x-x')a^{-1} + r$ (необходимо знать a чтобы найти r’)
- **Вычислительно связанный**
  - Если отправитель может найти различные x и x’ подойдут для обязательства $c=g^x h^r$, значит он может решить задачу дискретного логарифма
    - Если отправитель знает x,r,x’,r’ s.t. $g^x h^r = g^{x'} h^{r'}$
    - Так как $h=g^a \bmod |G|$, это значит $x+ar = x'+ar' \bmod |G|$
    - Отправитель может найти a вычислив $(x'-x)(r-r')^{-1}$

ОДНОПРОХОДНАЯ СХЕМА ГОЛОСОВАНИЯ

Схема

$P_1: v_1$

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

$P_2: v_2$

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

$P_n: v_n$

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

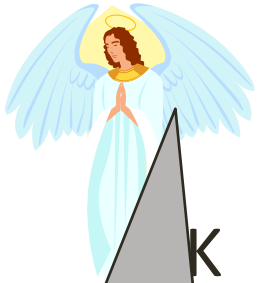
PK

BB

C_1

C_2

C_n



K

Использует SK для вычисления $v_1,\ldots v_n$. Вычисляет и возвращает $\rho(v_1,v_2,\ldots,v_n)$

Описание схемы

- **Setup(v)**: построить (**x**,**y**,**BB**) секретную информацию для подсчета голосов, публичные параметры схемы, инициализации **BB**
- **Vote(y,v)**: алгоритм, исполняемый каждым голосующим, для получения бюллетеня **b**
- **Ballot(BB,b)**: выполняется доской голосования; возвращает новые **BB** и да/нет
- **Tallying(BB,x)**: исполняется ЦИК и возвращает результат голосования

Реализация: Enc2Vote

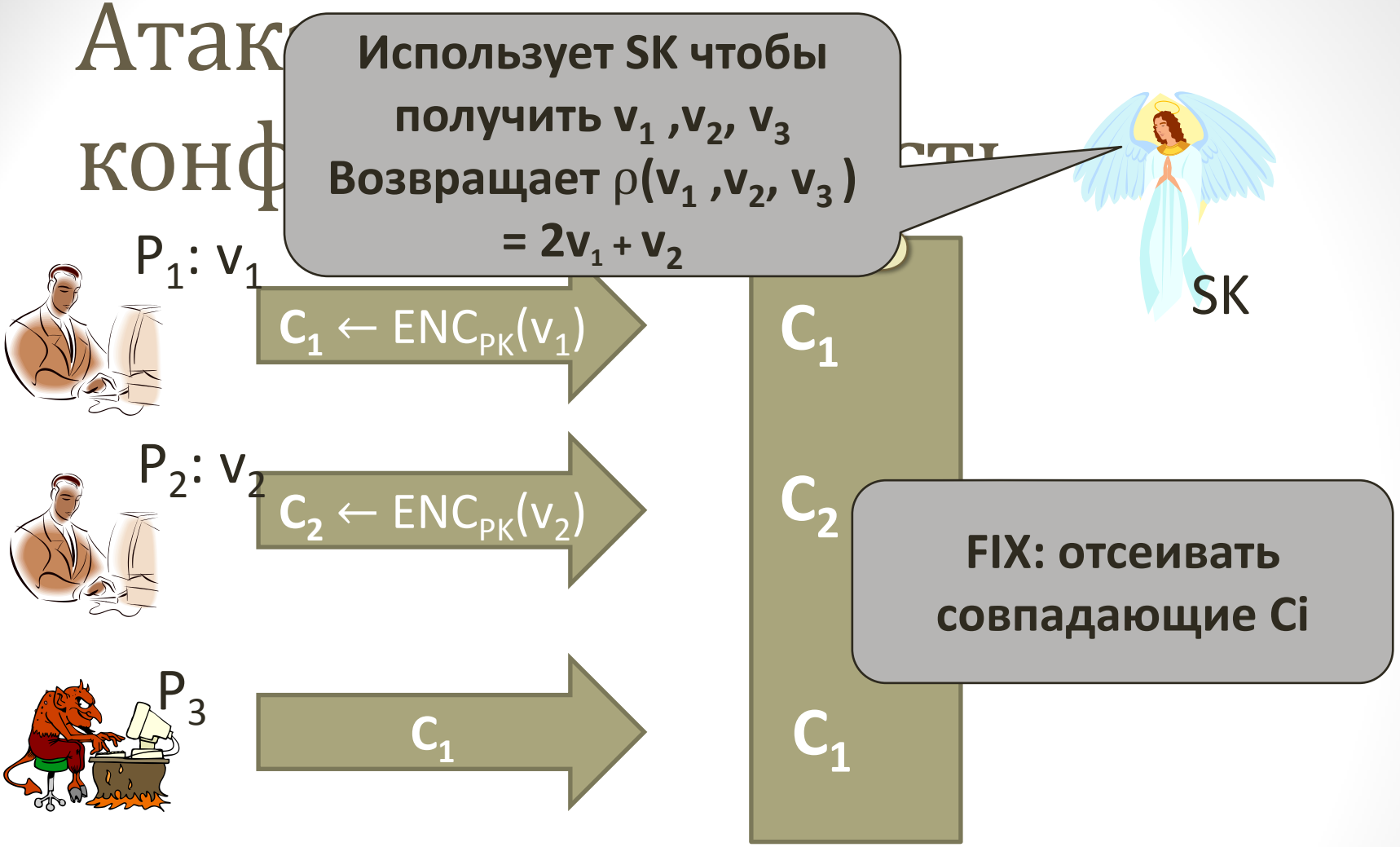
- Пусть π=(KG,ENC,DEC) гомоморфная схема шифрования. Enc2Vote(π) :
- **Setup(v)**: KG генерирует (SK,PK,[])
- **Vote(PK,v)**: b ← $ENC_{PK}(v)$
- **Process Ballot([BB],b)**: [BB] ← [BB,b]
- **Tallying([BB],x)**: где [BB] = $[b_1,b_2,\ldots,b_n]$

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

  - **result** ←$DEC_{SK}(x,b)$

    возвращает **result**

Атак… конф…сти



- При условии, что значения v 0 или 1
- Если результат голосования 0 или 1 значит v_1=0, иначе v_1 =1

Нова

По SK вычисляет v_1 ,v_2, v_3 Возвращает $\rho(v_1, v_2, v_3) = 2v_1 + v_2$

SK

P_1: v_1

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

P_2: v_2

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

P_3

C

C_1

C_2

C

FIX: Убедитесь, что принятые шифротексты не являются скрытыми или измененными копиями

Вычисляет
$C_0=ENC_{PK}(0)$
И $C=C_1 \cdot C_0=ENC_{PK}(v_1)$

Non-malleable encryption (NM-CPA => SS-CPA)

Nonnmalleability of $\pi = (\text{Setup}, \text{Kg}, \text{Enc}, \text{D}$

Public Key

PK

M_0, M_1

C

$C_1, C_2 \ldots, C_n$

$M_1, M_2, \ldots, M_n$

Guess d

π

Params ← Setup()

(PK,SK) ← Kg (params)

b ← {0, 1}

C ← $\text{Enc}_{PK}(M_b)$

M_i ← $\text{Dec}_{PK}(C_i)$, for i=1..n

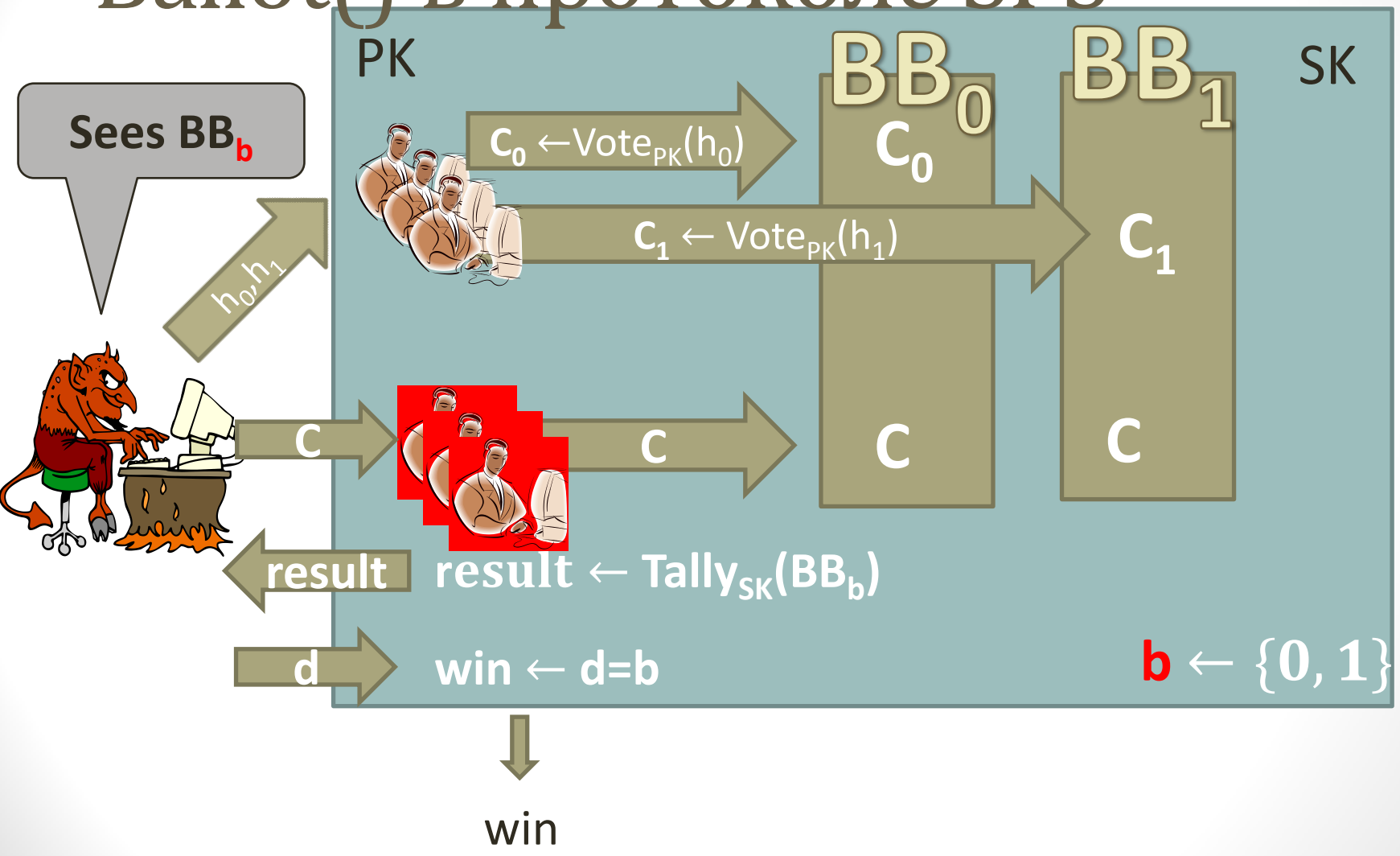
win ← d=b

win

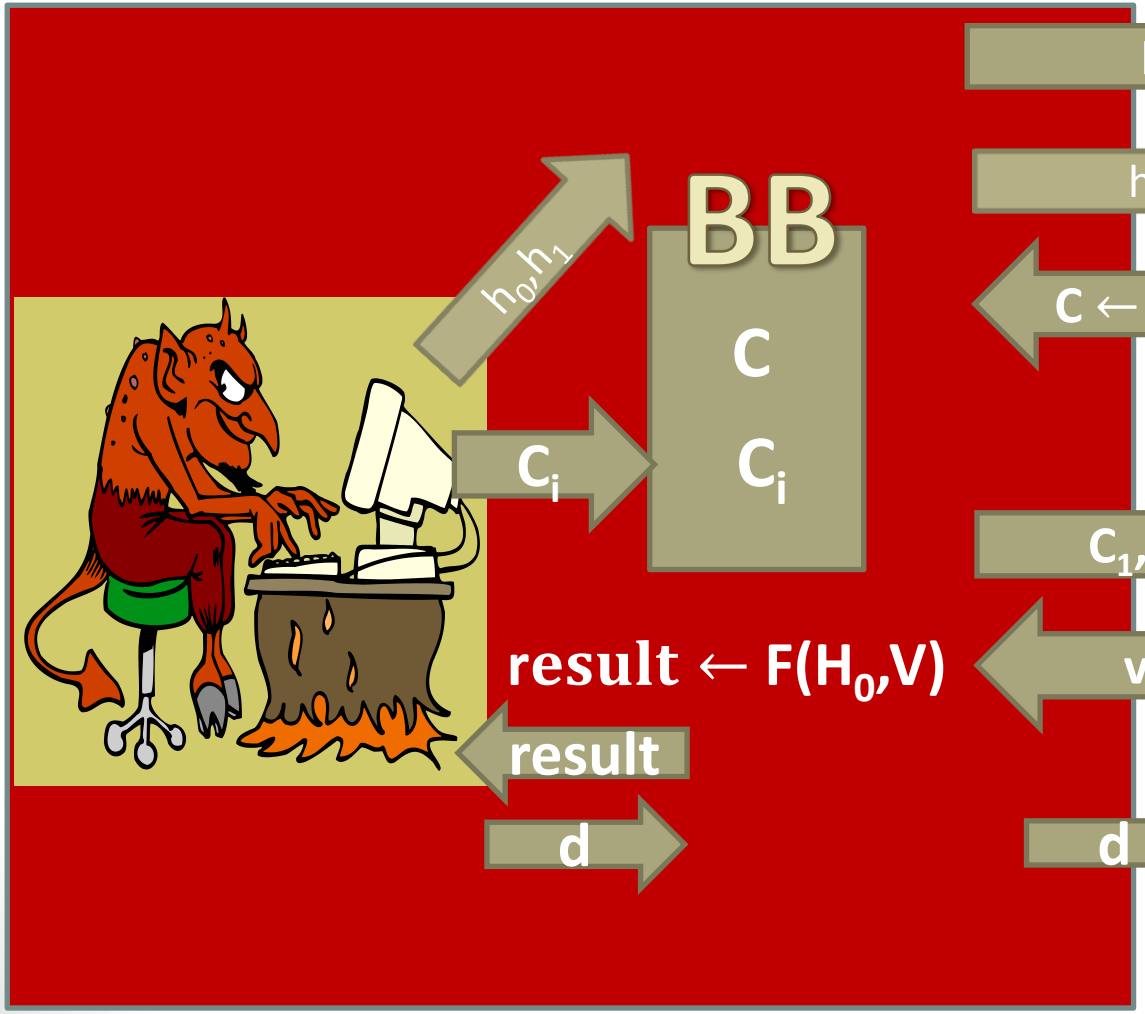
ElGamal is not non-malleable

- Любая схема гомоморфного шифрования нестойкая (NM-CPA):
  - Для любого $Enc_{PK}(m)$ легко можно вычислить $Enc_{PK}(m+1)$ (умножив на зашифрованную 1)
- Для ElGamal:
  - Выберем пару сообщений 0,1
  - Получим c=(R,C)
  - Запросим расшифровку для(R,C·g). Если ответ 1, то b = 0, а если 2, то b = 1

Стойкость процедуры Ballot() в протоколе SPS



Theorem: If π is a non-malleable encryption scheme then Env2Vote(π) has vote secrecy.



Шифрование Paillier

- Публичный ключ $N=PQ=(2p+1)(2q+1)$
- Секретный ключ d такой что $d=1 \bmod N$, $d=0 \bmod 4pq$
- Шифрование голоса $v \in \mathbf{Z}_N$ на случайном $R \in \mathbf{Z}_N^*$

$$C = (1+N)^v R^N \bmod N^2$$

- Дешифрование

$$v = (C^d - 1 \bmod N^2)/N$$

Корректность

- Публичный ключ $N=PQ=(2p+1)(2q+1)$
- Секретный ключ d такой что $d=1 \bmod N$, $d=0 \bmod 4pq$
- Размер мультипликативной группы $\mathbf{Z}_{N^2}^*$ равен $4Npq$
- При этом $(1+N)^N = 1 + N \cdot N + \ldots \equiv 1 \bmod N^2$
- Проверка

$$C^d = ((1+N)^v R^N)^d = (1+N)^{vd} R^{Nd}$$

$$= (1+N)^{vd} R^{4Npqk} \equiv (1+N)^v \bmod N^2$$

$$(1+N)^v = 1+vN+\binom{v}{2}N^2+\ldots \equiv 1+vN \bmod N^2$$

$$(C^d-1 \bmod N^2)/N = v$$

Гомоморфность

- Публичный ключ $N=PQ=(2p+1)(2q+1)$
- Шифрование голоса $v \in \mathbf{Z}_N$ на случайном $R \in \mathbf{Z}_N^*$

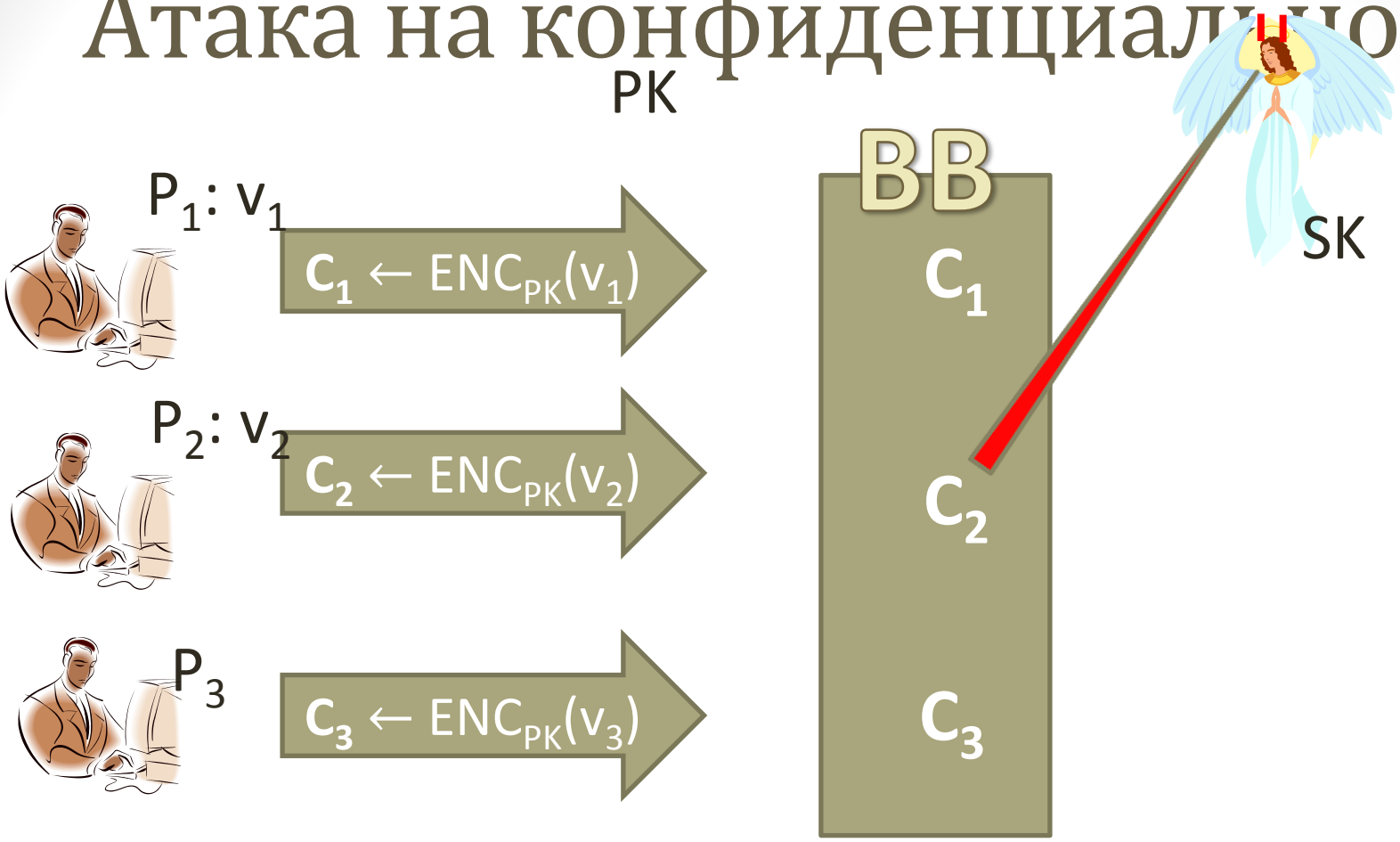
  $C = (1+N)^v R^N \bmod N^2$

- Гомоморфизм

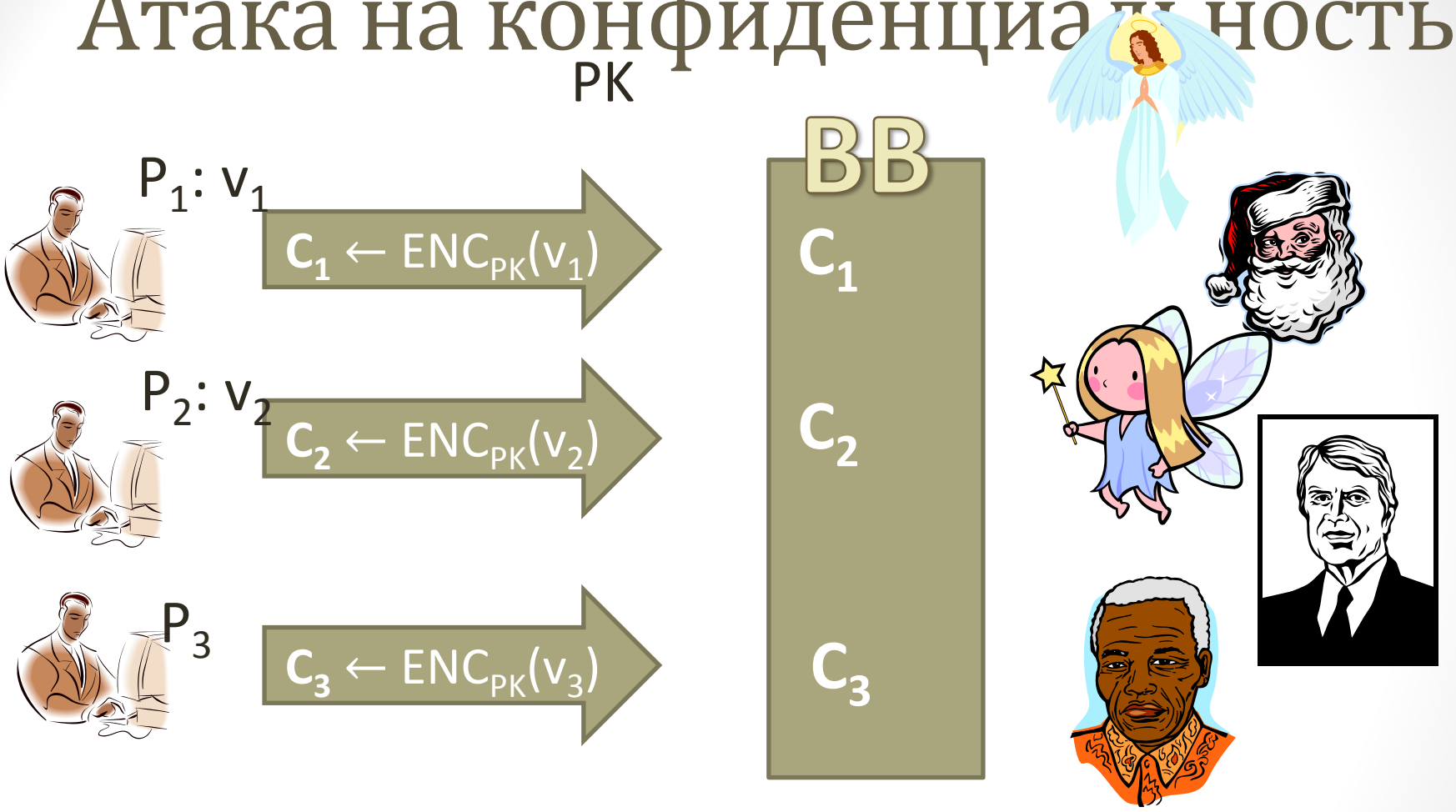
$$(1+N)^v R^N \cdot (1+N)^w S^N$$

$$\equiv \quad (1+N)^{v+w}(RS)^N \bmod N^2$$

Атака на конфиденциальность

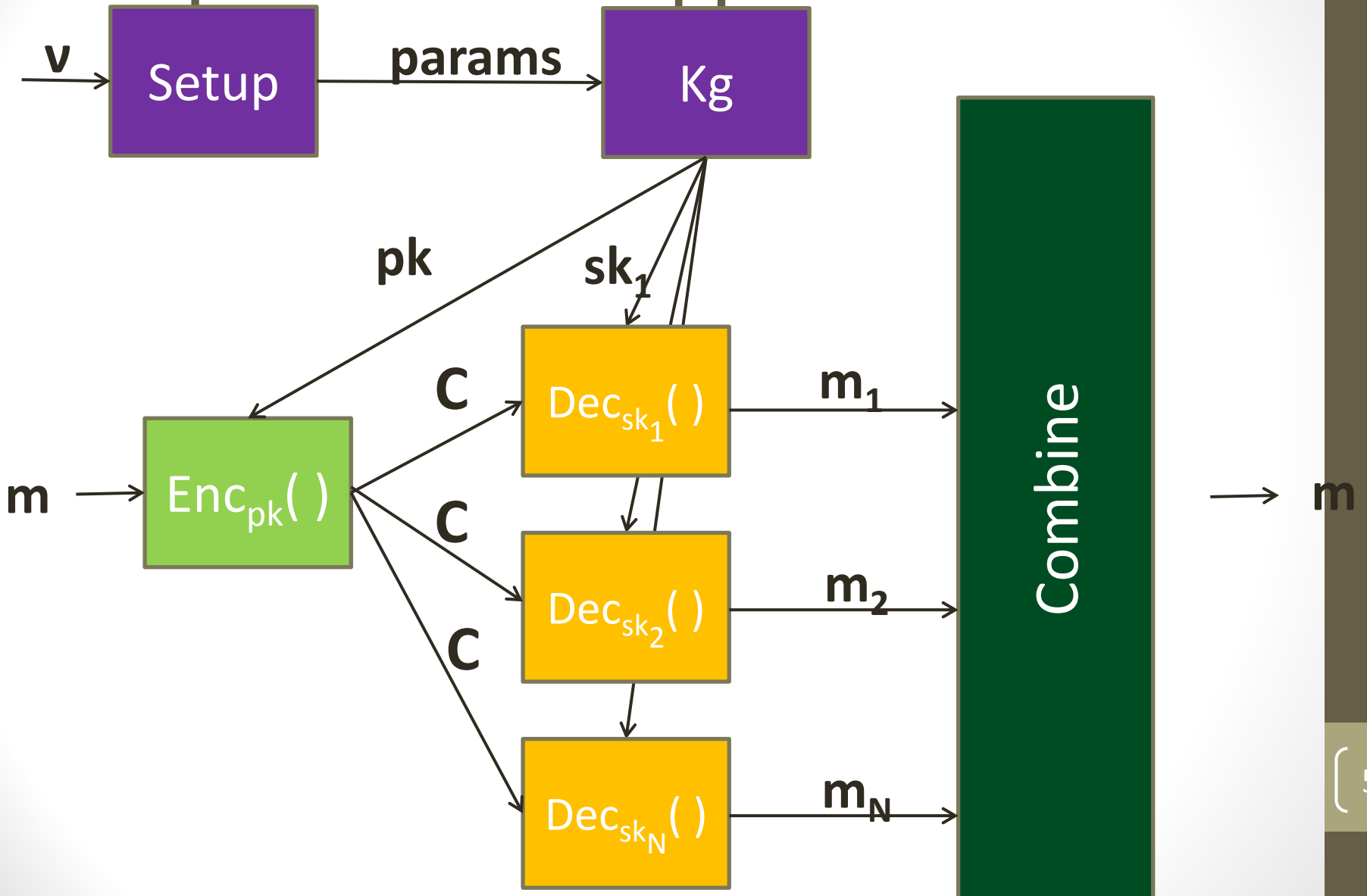


Атака на конфиденциальность



ПОДСЧЕТ ГОЛОСОВ БЕЗ НАРУШЕНИЯ КОНФИДЕНЦИАЛЬНОСТИ

Пороговое шифрование



Пороговое шифрование

- **Описание:**
  - **Key Generation(n,k)**: outputs pk,vk,(sk_1, sk_2, …,sk_n)
  - **Encrypt(pk,m)**: outputs a ciphertext C
  - **Decrypt(C,sk_i)**: outputs m_i
  - **ShareVerify(pk,vk,C, m_i)**: outputs accept/reject
  - **Combine(pk,vk,C,{m_{i1},m_{i2},…,m_{ik}})**: outputs m

ElGamal

- **Setup(v)**: выбрать м. группу $(G,\cdot)$ и ее генератор g
- **KG(G, g)**: $x \leftarrow \{1,\ldots,|G|\}$;
  $X \leftarrow g^x$
  output (X,x)
- **$ENC_X(m)$**: $r \leftarrow \{1,\ldots,|G|\}$;
  $(R,C) \leftarrow (g^r, g^m X^r)$;
  output (R,C)
- **$DEC_x((R,C))$**: найти t такое что $g^t = C/R^x$
  output m=t

(k,n) пороговая схема ElGamal

- Генерация ключей :
  - $s_1,s_2,\ldots,s_n$ по схеме Шамира.
  - Публичный ключ $X=g^s$ the verification key ключи проверок $X_1=g^{s1}$, $X_2=g^{s2},\ldots,X_n=g^{sn.}$.
  - Участнику i выдают $s_i=P(i)$
- Частичное дешифрование $(s_i,(R,C))$:
  - Сторона i вычисляет $m_i=R^{si}$
- Сбор$((R,C),m_1,\ldots,m_N)$:

  $R^s = R^{P(0)} = R^{\Sigma s_i \Pi (-j)/(i-j)} = \prod R^{s_i c_i}$

  где $c_j=\Pi (-j)/(i-j)$ (произведение по $i \in I-\{j\}$)

Mixnets

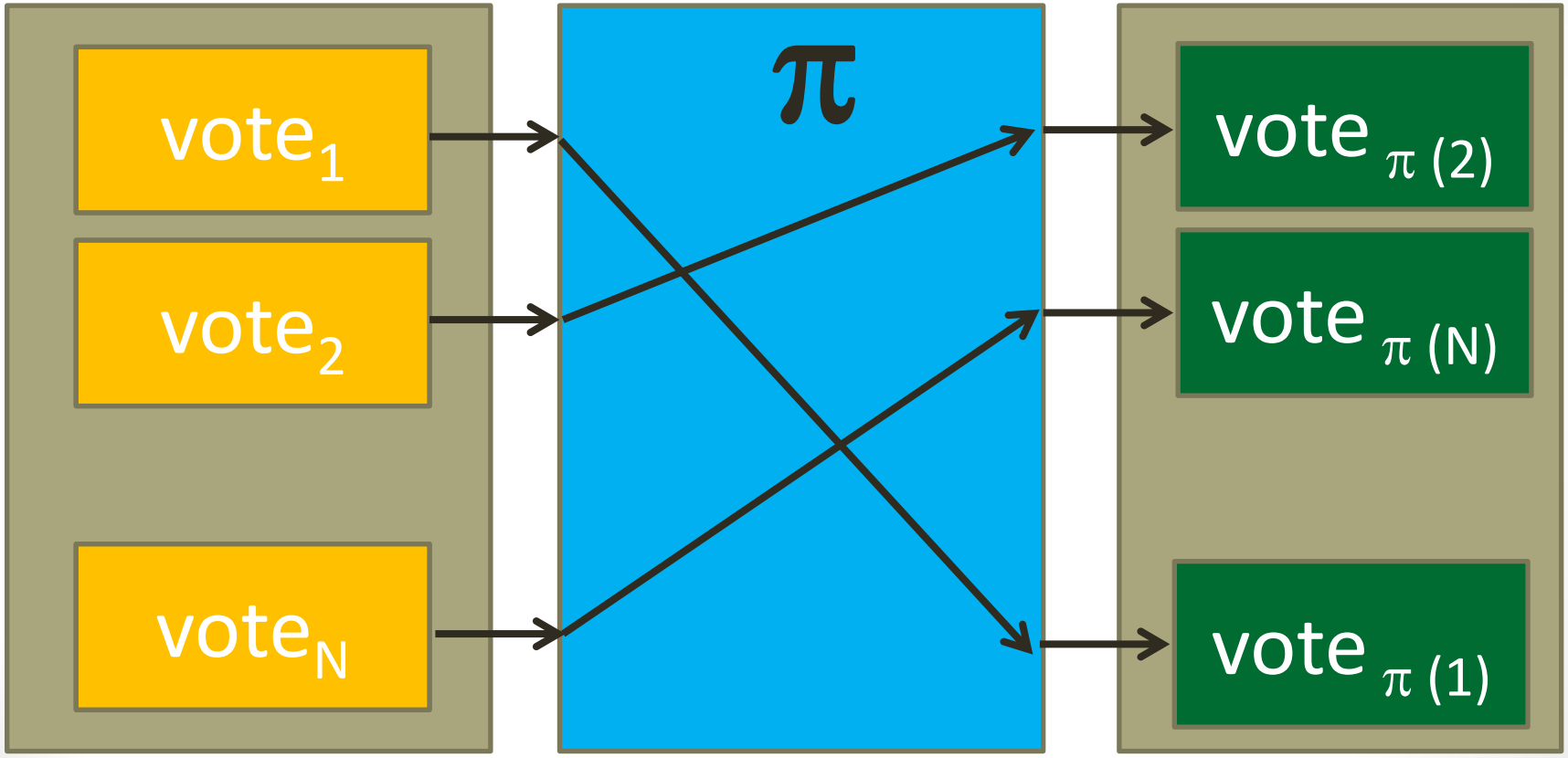
- Использование гомоморфных ф-ций сложно для сложных ф-ций
  - Вместо этого $Enc_{pk}(f(v_1,v_2,\ldots,v_n))$ можно просто расшифровать голоса

Дополнительная рандомизация

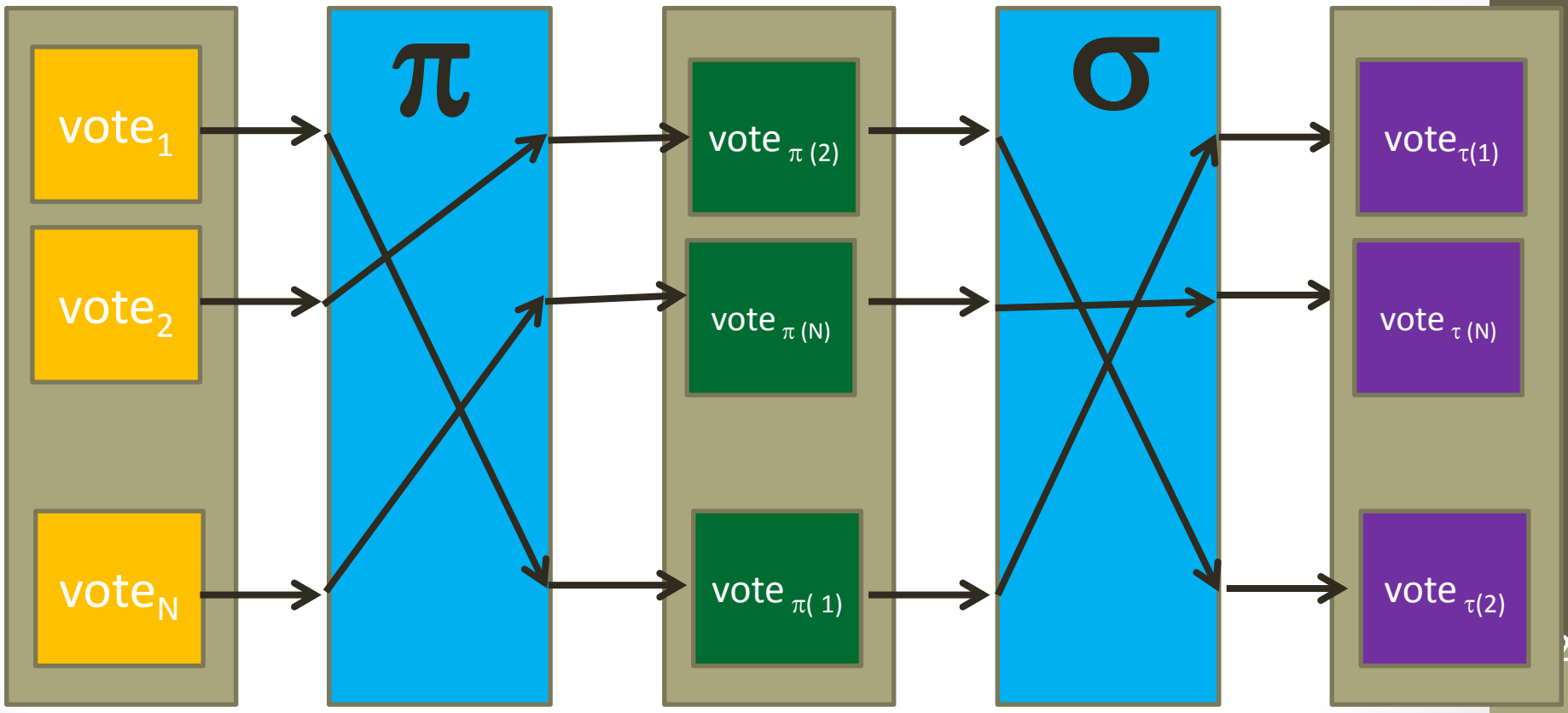
$$\mathbf{Enc_{pk}(m;r) \bullet Enc_{pk}(0;s) = Enc_{pk}(m;r+s)}$$

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

Mixnet

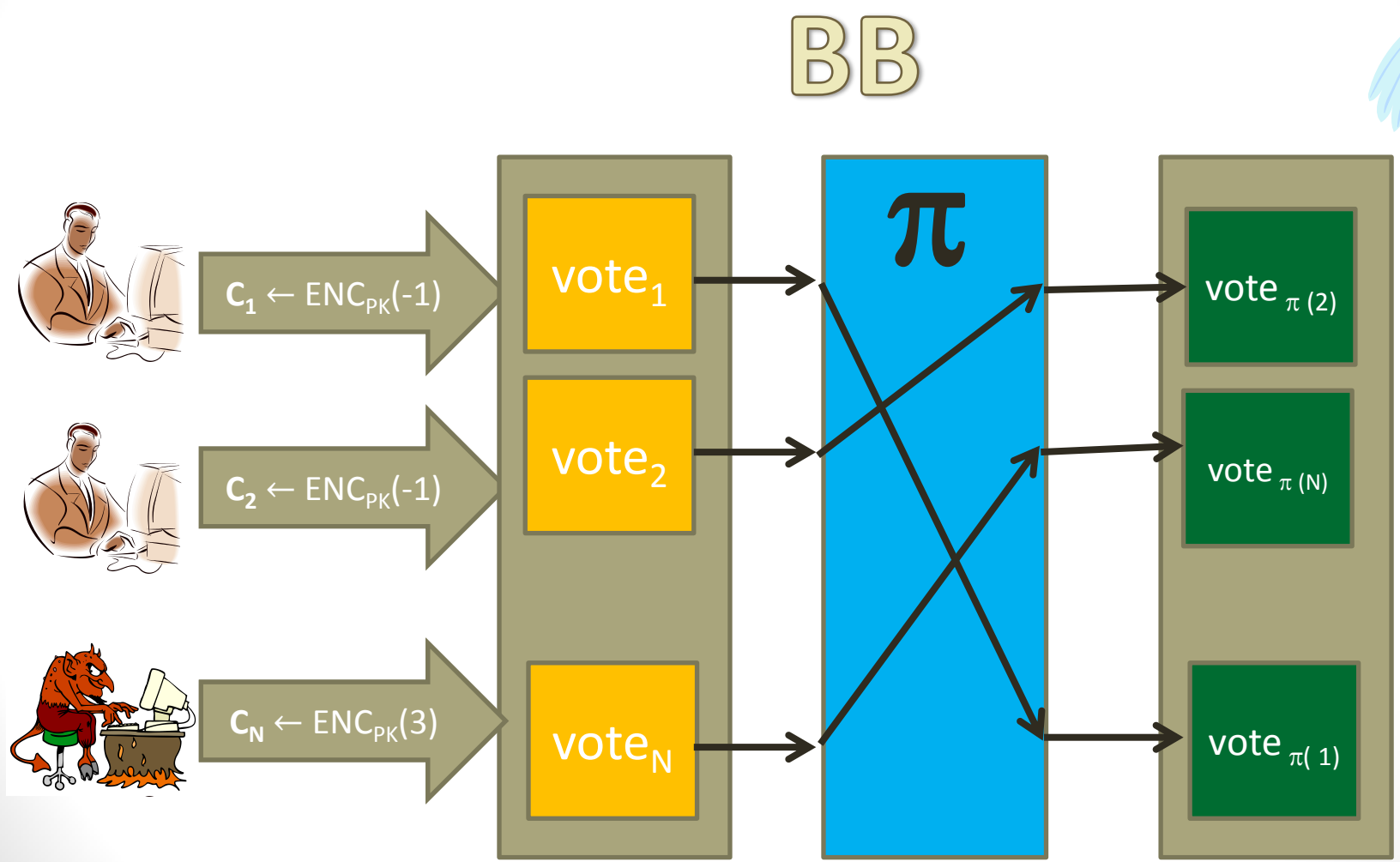


Mixnet

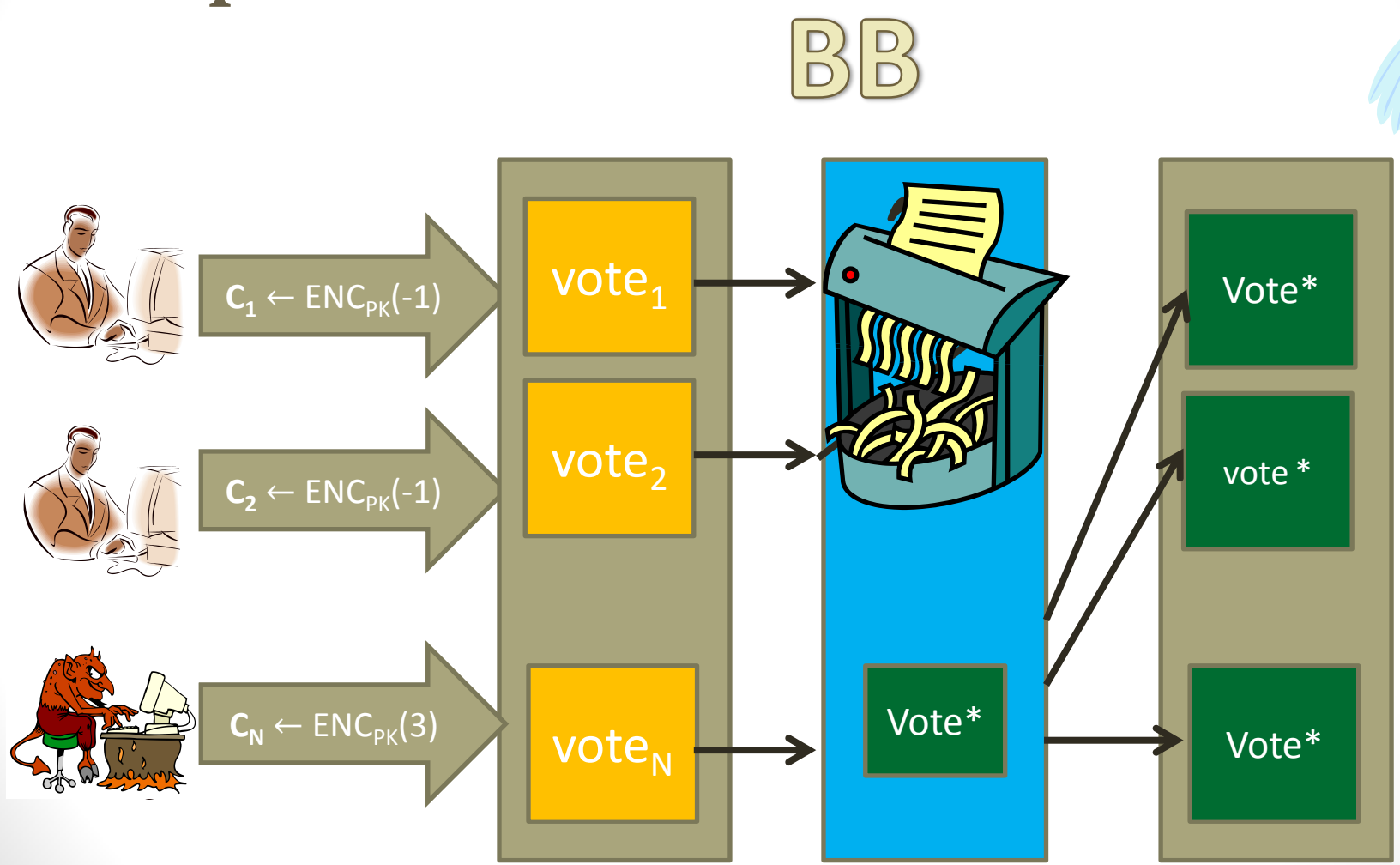


$\tau = \pi ; \sigma$

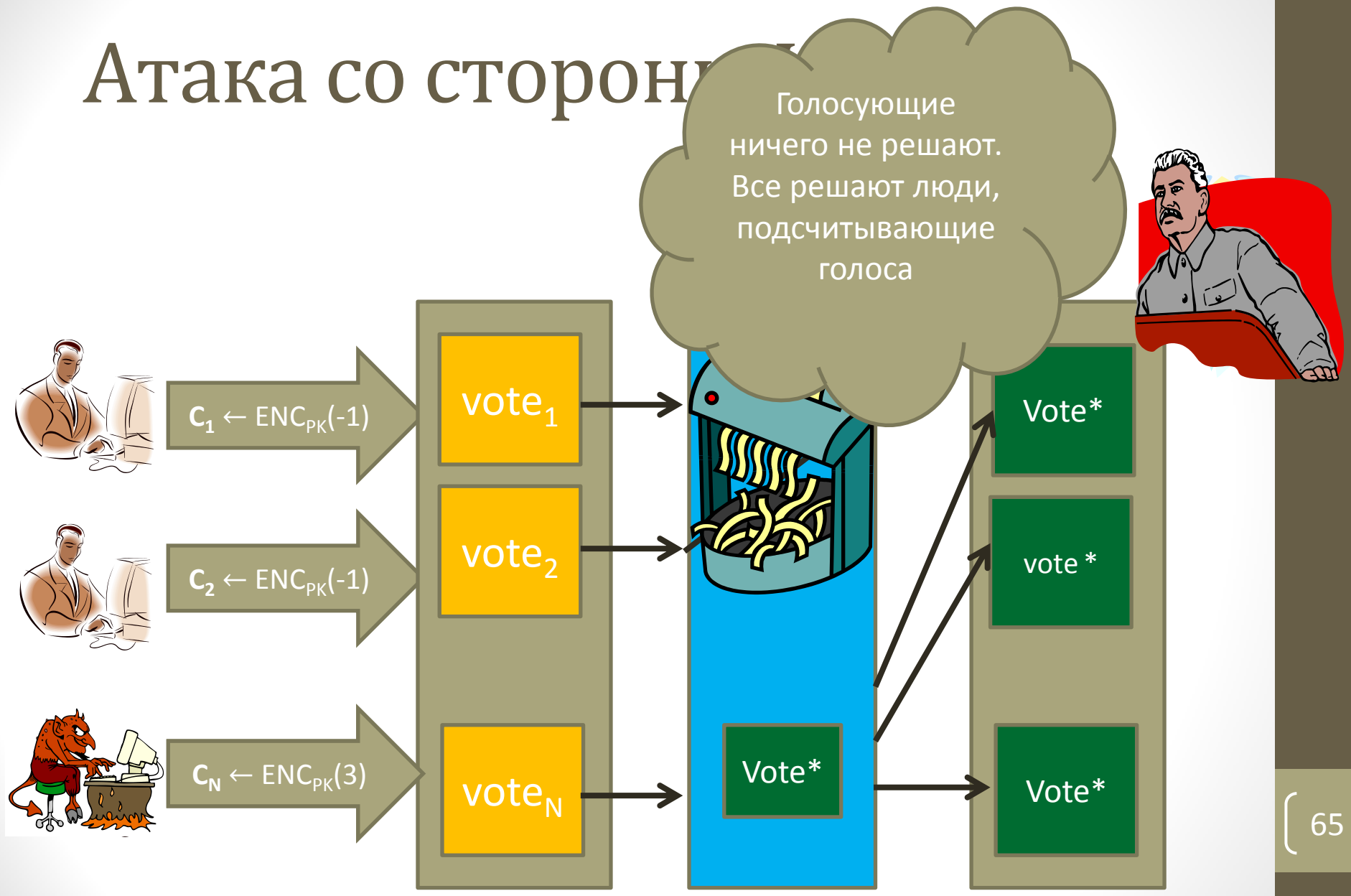
Атака со стороны голосующих



Атака со стороны перестановочной сети



Атака со сторон



Угрозы

- **Голосующие**: плохо отформатированные голоса; проблематично для гомоморфного подсчета
- **Перемешивающий сервер**: может полностью подменить бюллетени
- **ЦИК**: может врать про результаты подсчетов

ДОКАЗАТЕЛЬСТВА С НУЛЕВЫМ РАЗГЛАШЕНИЕМ

Интерактивное доказательство[GMW91]

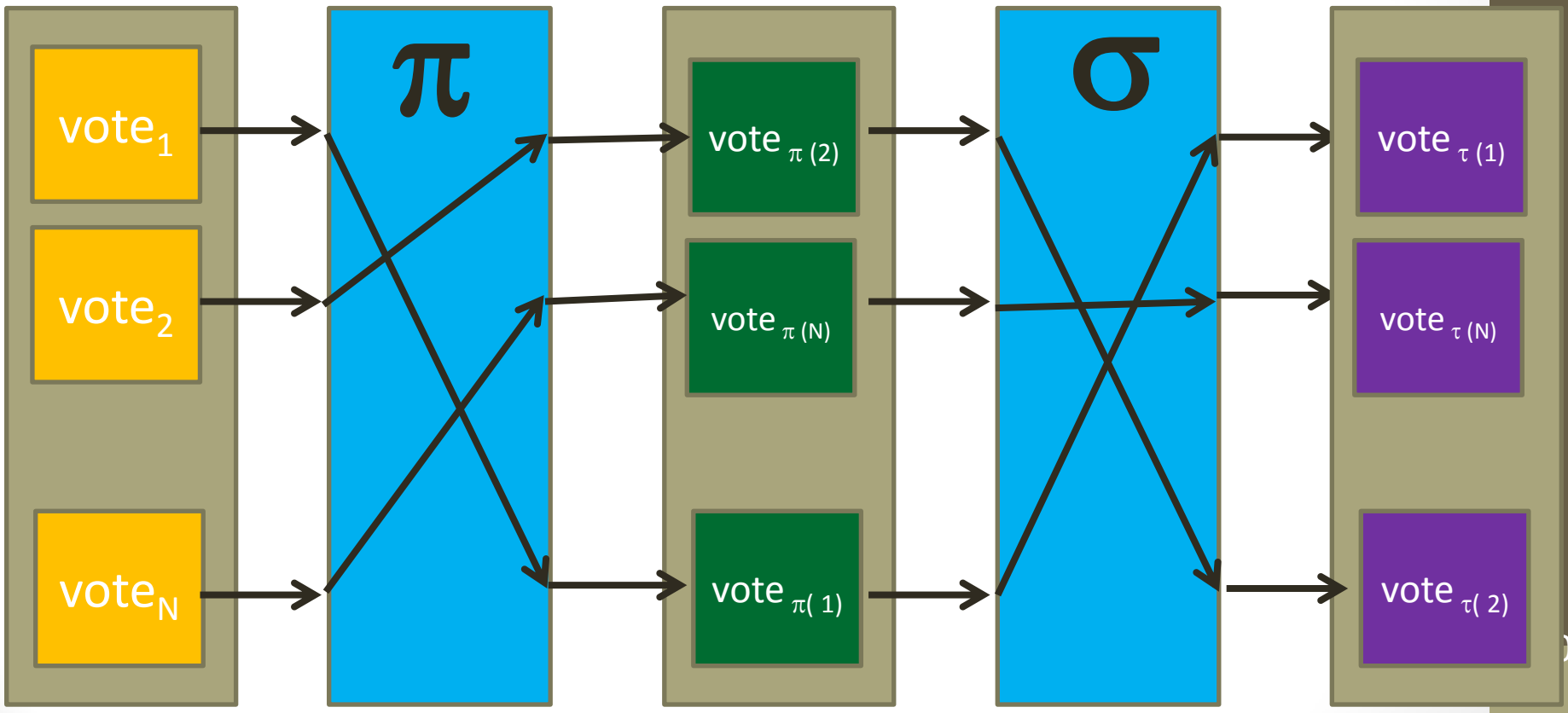
Примеры:

- $Rel_{g,h}((X,Y),z)$ iff $X=g^z$ and $Y=h^z$
- $Rel_{g,X}((R,C),r)$ iff $R=g^r$ and $C=X^r$
- $Rel_{g,X}((R,C),r)$ iff $R=g^r$ and $C/g=X^r$
- $Rel_{g,X}((R,C),r)$ iff $(R=g^r$ and $C=X^r)$ or $(R=g^r$ and $C/g=X^r)$
- $Rel_L(X,w)$ iff $X \in L$

Свойства

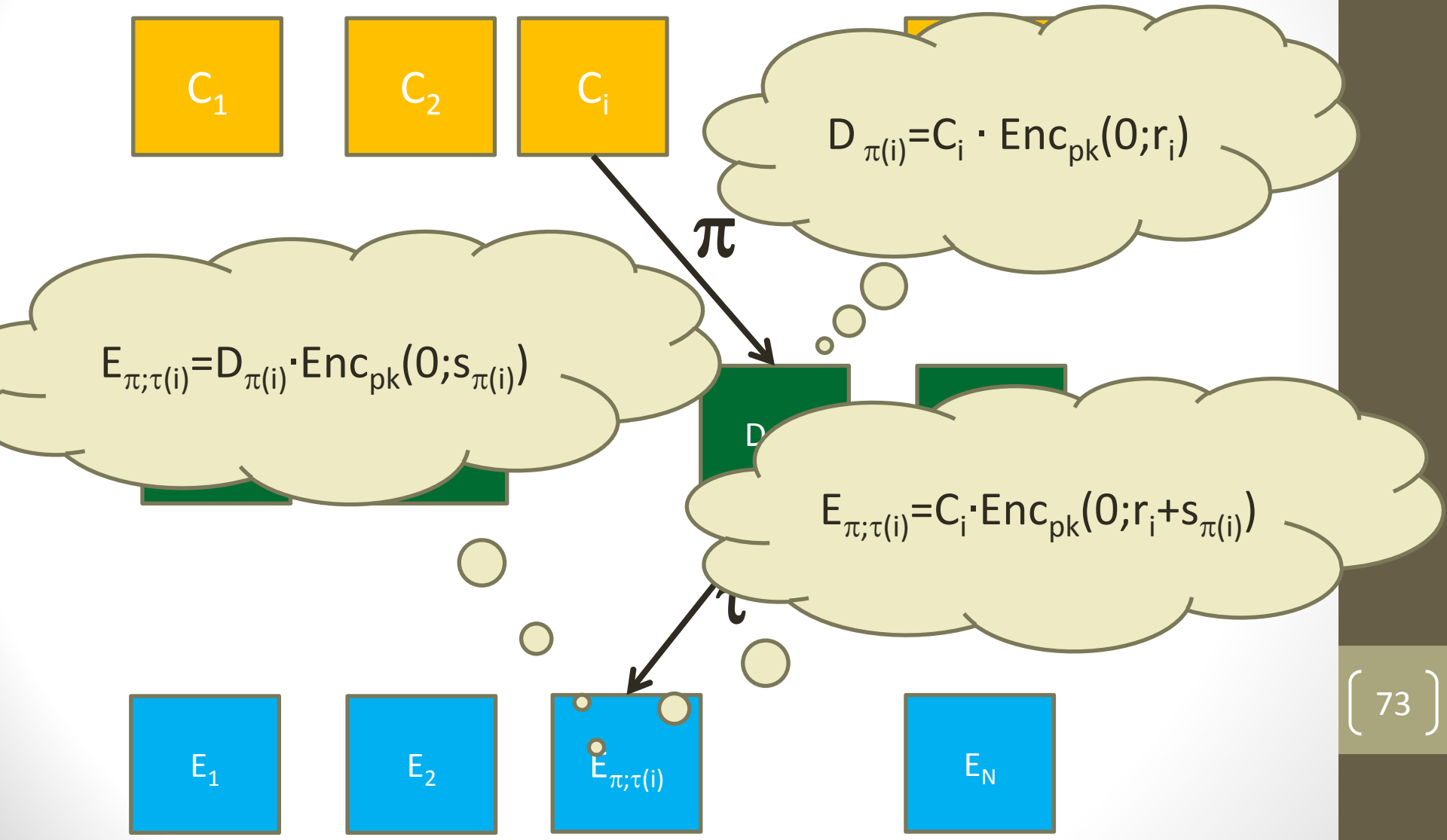
- **Полнота**: честный Prover всегда докажет честному Verifier правильность утверждения
- **Soundness**: нечестный Prover может обманывать только с очень малой вероятностью
- **Zero knowledge**: Никакая дополнительная информация не разглашается
- **Proof of knowledge**: из успешной проверки можно извлечь свидетеля

Mixnet



$\tau = \pi ; \sigma$

Проверяемая перестановка



Проверяемая перестановка[KS95]

- Prover знает $C_1,C_2,\ldots,C_n$, $D_1,D_2,\ldots,D_n$, перестановку π и случайные монеты $r_1,r_2,\ldots,r_n$ такие что $D_i=C_{\pi(i)} \cdot Enc_{pk}(0;r_i)$

- Prover выбирает перестановку τ и случайные $s_1,s_2,\ldots,s_n$ и потом он вычисляет и отправляет $\{E_{\pi;\tau(i)}=D_{\pi(i)} \cdot Enc_{pk}(0; s_{\pi(i)})\}_i$
- Verifier выбирает случайный бит b и отправляет его Prover
- Prover отвечает следующим образом
  - If b=0 then посылает $(\pi;\tau)$ и $r_1+s_{\pi(1)}$
  - If b=1 then посылает τ, $s_1,s_2,\ldots,s_n$
- Когда verifier получил σ, $q_1,q_2,\ldots q_n$ он проверяет равенства:
  - If b=0: $E_{(\pi;\tau)(i)}=C_i \cdot Enc_{pk}(0;r_i)$
  - If b=1: $E_{\tau(i)}=D_i \cdot Enc_{pk}(0;r_i)$

Helios


helios
IACR Elections 2012
public election created by David Pointcheval
Share
Tweet
Election of the IACR Directors
questions (1) | voters & ballots | trustees (3)
This election is complete.
Tally
Question #1
Director
Thomas Peyrin 166
Anna Lysyanskaya 270
Thomas Berson 226
Michel Abdalla 242
Xavier Boyen 157

Helios: Подготовка голоса

P: v

C

π

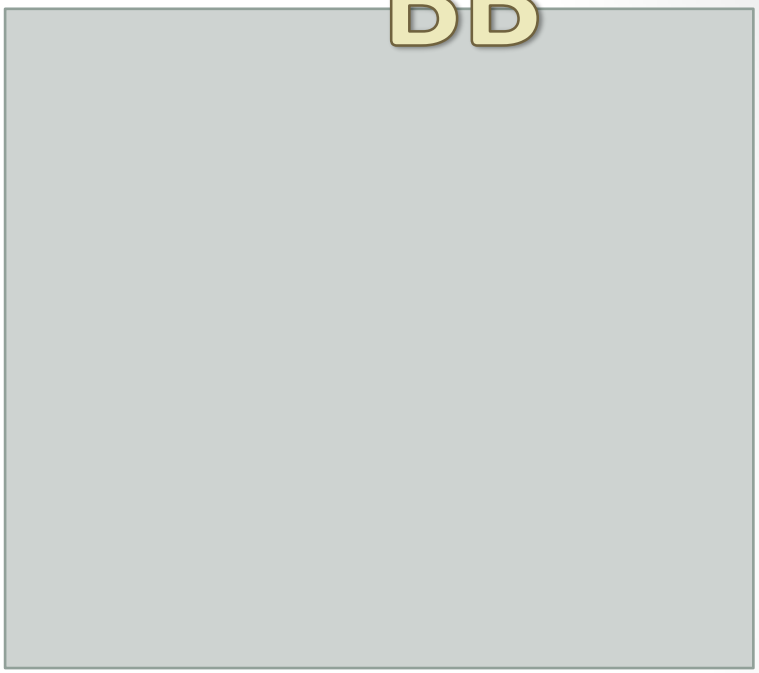
- $C = ENC_{PK}(v)$ шифрование голоса на публичном ключе для данных выборов
- π доказательство, что C верно

Helios: voting

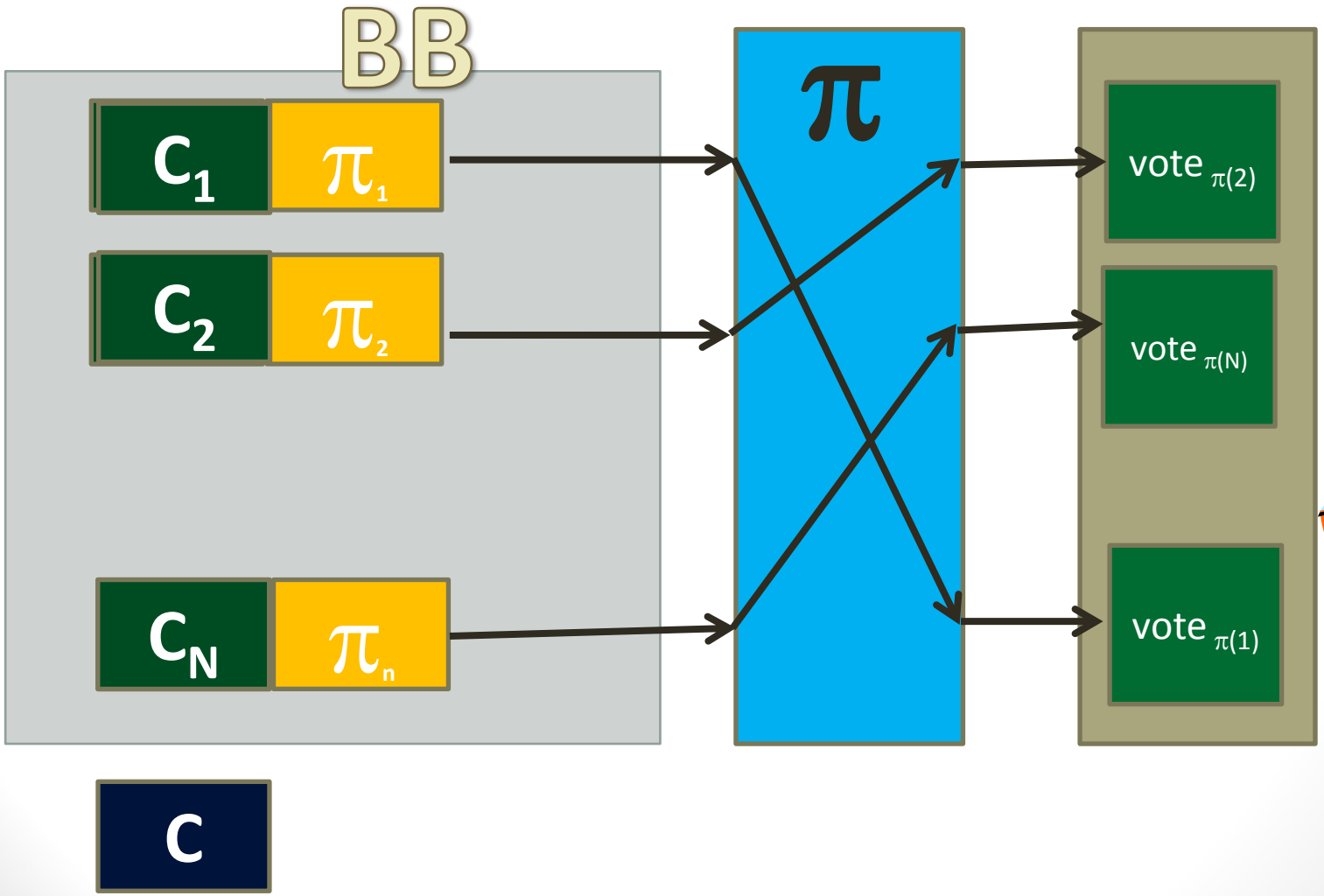
P_1: v_1

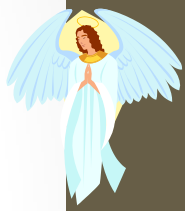
P_2: v_2

P_n: v_n



Helios: Подсчет голосов



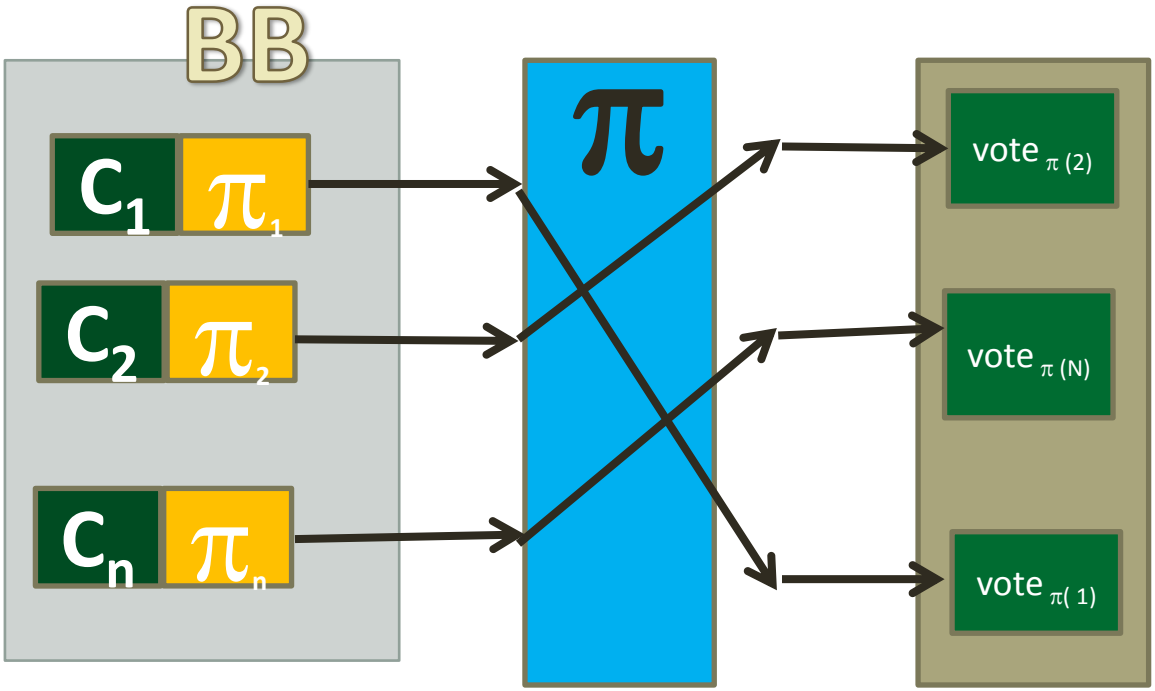


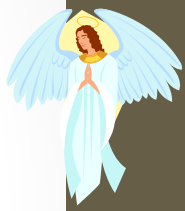
Helios


$P_1: v_1$


$P_2: v_2$


$P_n: v_n$





Электронные платежи

- Преимущества электронных денег
    - Дешевизна банковских операций
    - Анонимность
    - Защищенность от подделки
    - Возможность использования в электронном бизнесе

Требования и характеристики платежных систем

- Безопасность
  - Невозможность подделки
  - Невозможность превысить кредит
  - Невозможность двойной траты
  - Анонимность
  - Аппаратная/криптографическая основа стойкости

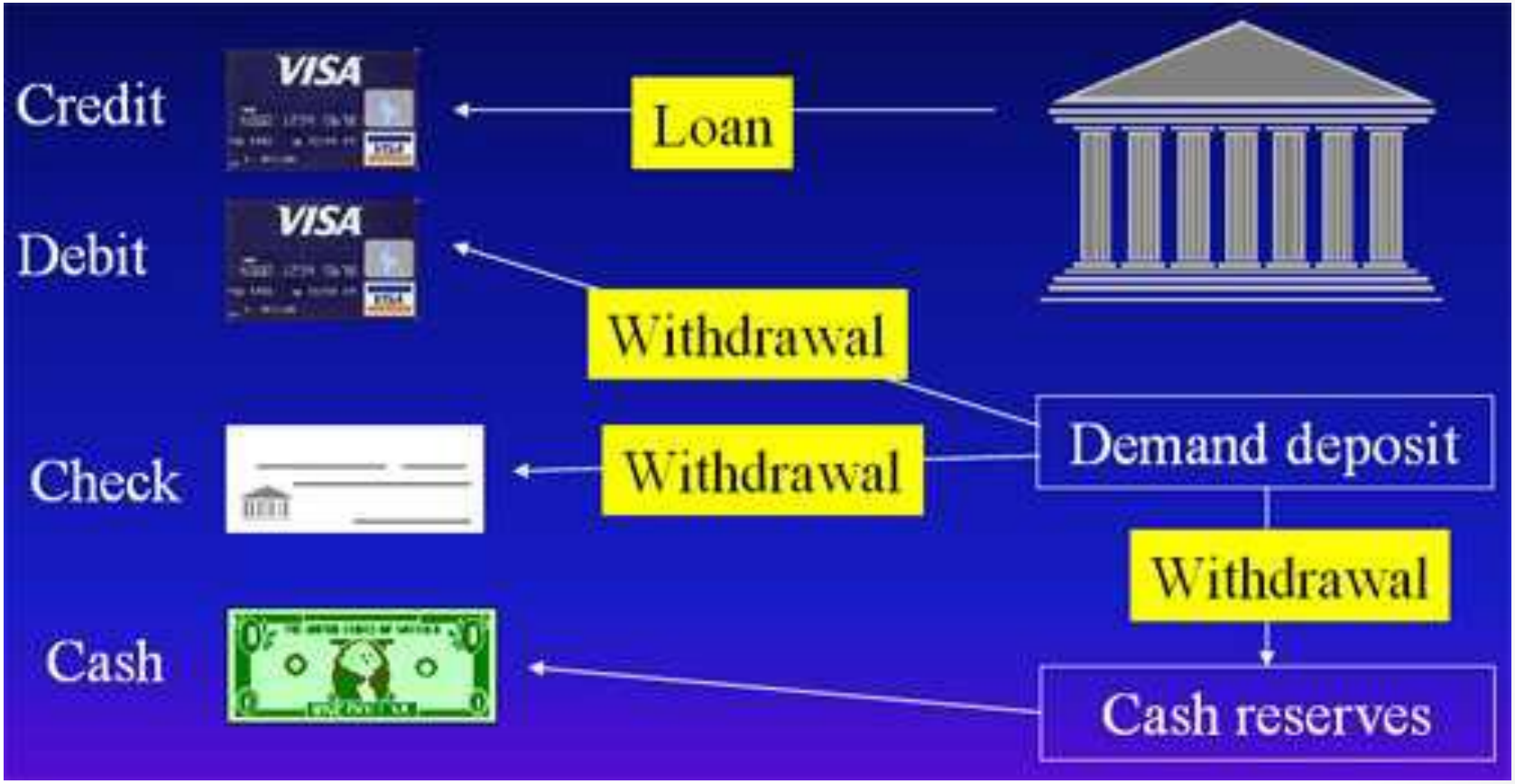
Требования и характеристики платежных систем

- Эффективность
  - Масштабируемость
  - Вычислительная трудоемкость
  - Коммуникационная сложность
  - Стоимость банковского обслуживания

Возможности платежных систем

- Возможности
  - Переносимость
  - Делимость
  - Универсальность
  - Возможность удаленной оплаты

Типы платежей

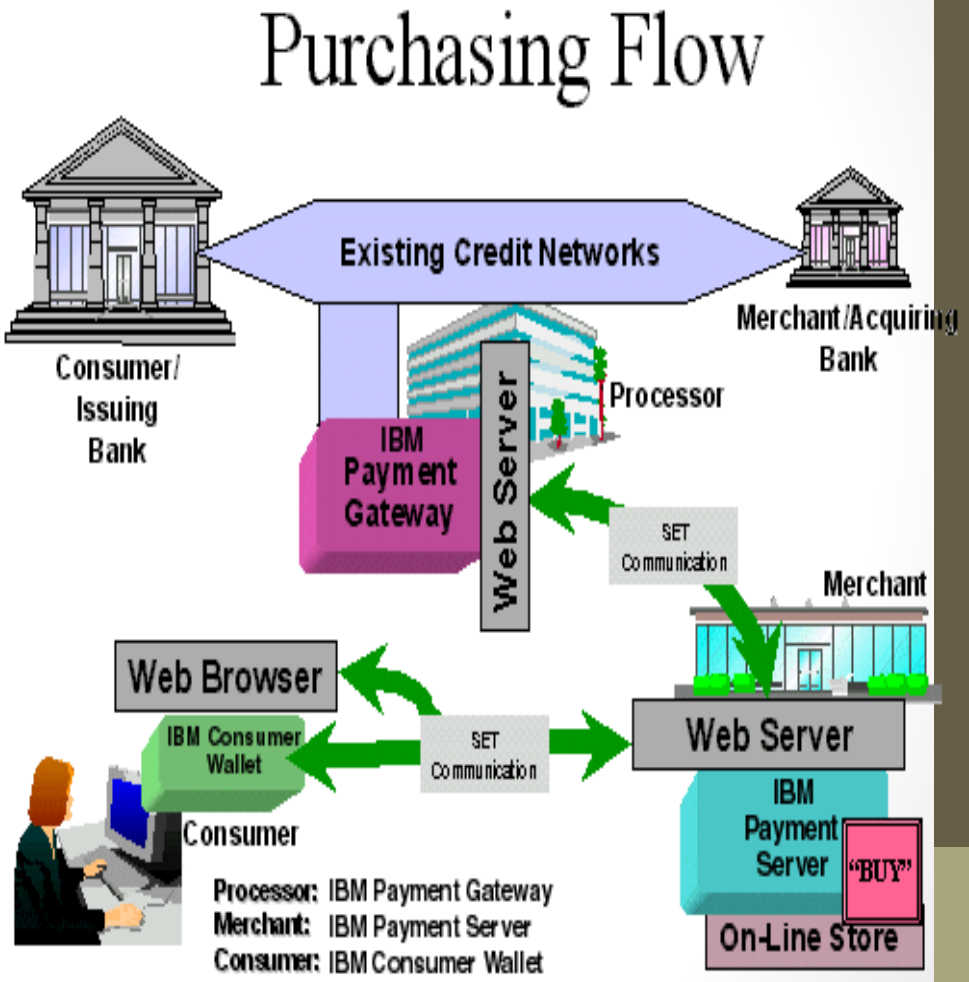


Кредитные карты

- Дебетные карточки
  - Кладем деньги
  - Сумма хранится на карточке
  - Потраченные деньги вычитаются из баланса карточки
- Кредитные карточки
  - Заводим счет в банке
  - Деньги снимаются со счета
- Криптографическая основа
  - Протокол SET

Secure Electronic Transaction (SET)

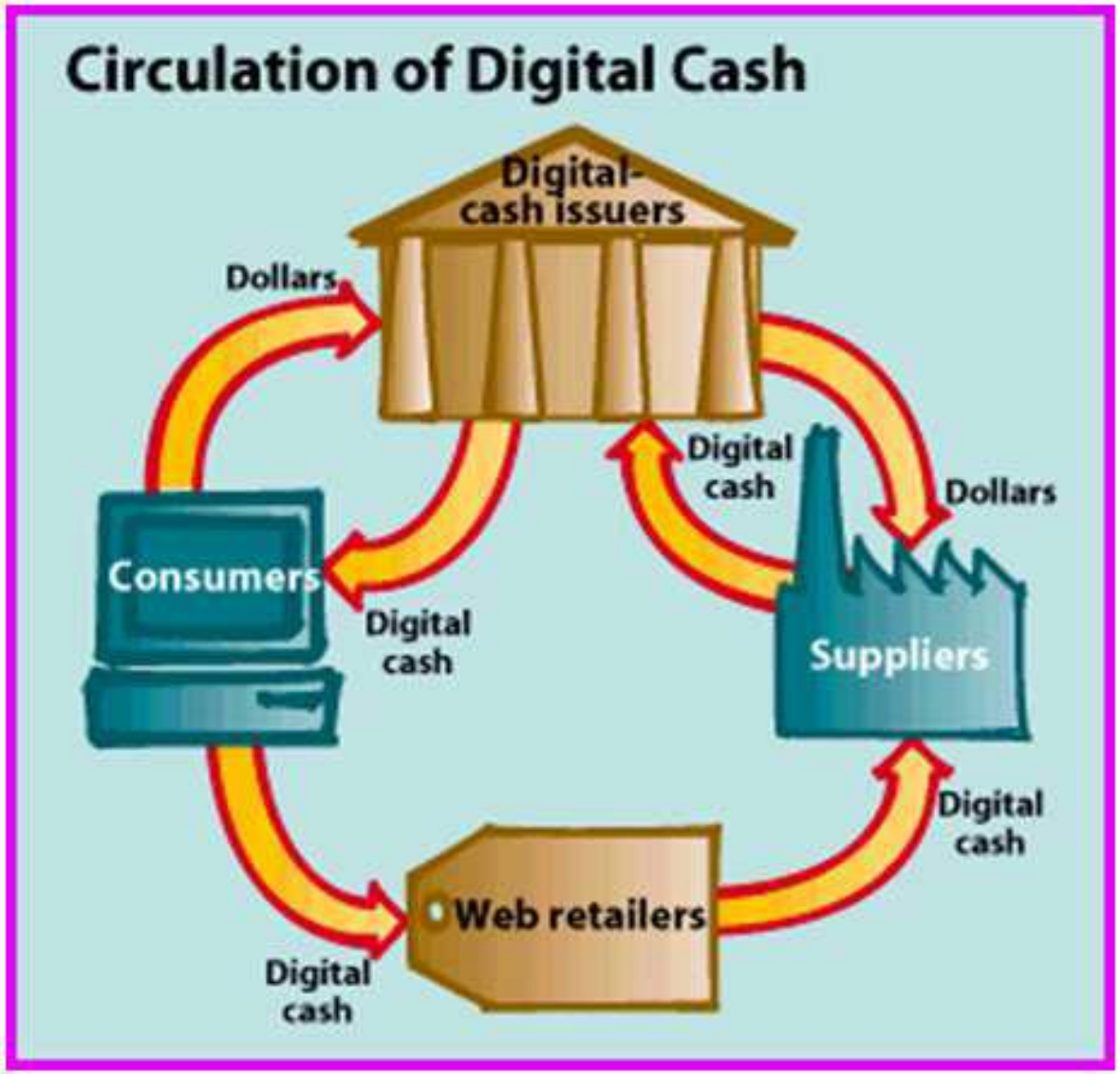
- Протокол разработан Visa International и MasterCard International
- SET использует цифровые сертификаты для удостоверения всех участников протокола
  - Покупателя, продавца, банки продавца и покупателя



Микроплатежи

- Специфика
  - Не требуется анонимность
  - Минимизация криптографических вычислений
- Современные решения
  - Millicent
  - Payword

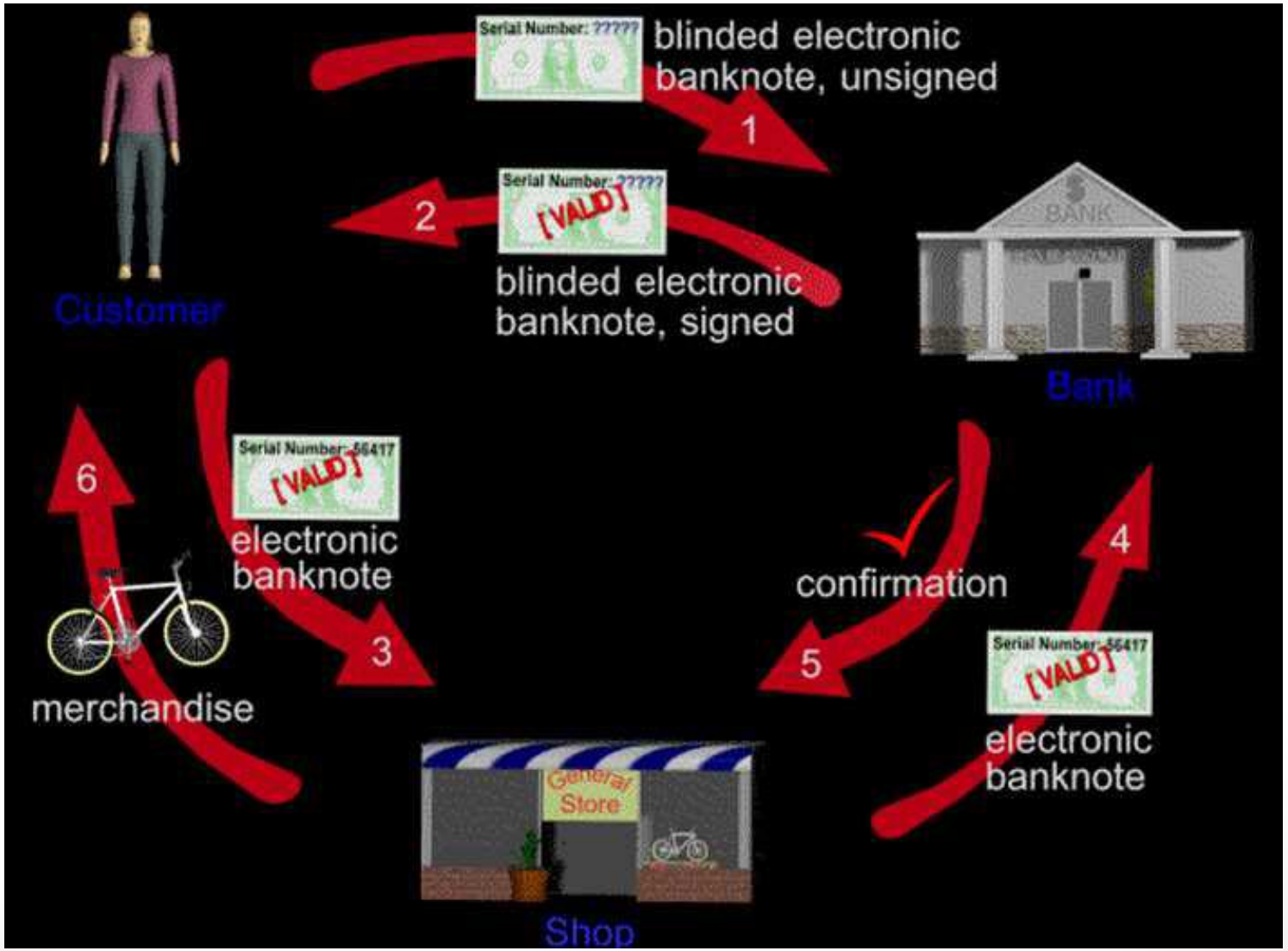
Электронные наличные



Используемая криптография

- Аутентификация сообщений
  - Гарантирует целостность
- Шифрование
  - Гарантирует секретность операций от посторонних
- Цифровые сертификаты
  - Защищают от мошенников
- Слепая подпись
  - Используется для электронных наличных (Аннонимность)

Общий вид схемы



Наивный протокол

- Обналичивание
  - 1) Участник просит Банк выдать 100$
  - 2) Банк присылает счет на 100$: { Я счет на 100$ #4257 }SK_B
  - 3) Участник проверяет подпись и признает счет
- Оплата
  - 1) Участник посылает продавцу счет
  - 2) Продавец проверяет подпись и признает счет
- Получение денег
  - 1) Продавец посылает счет в Банк
  - 2) Банк проверяет свою подпись и переводит деньги продавцу

Недостатки

- Можно тратить дважды
- Нет анонимности

Построение анонимности

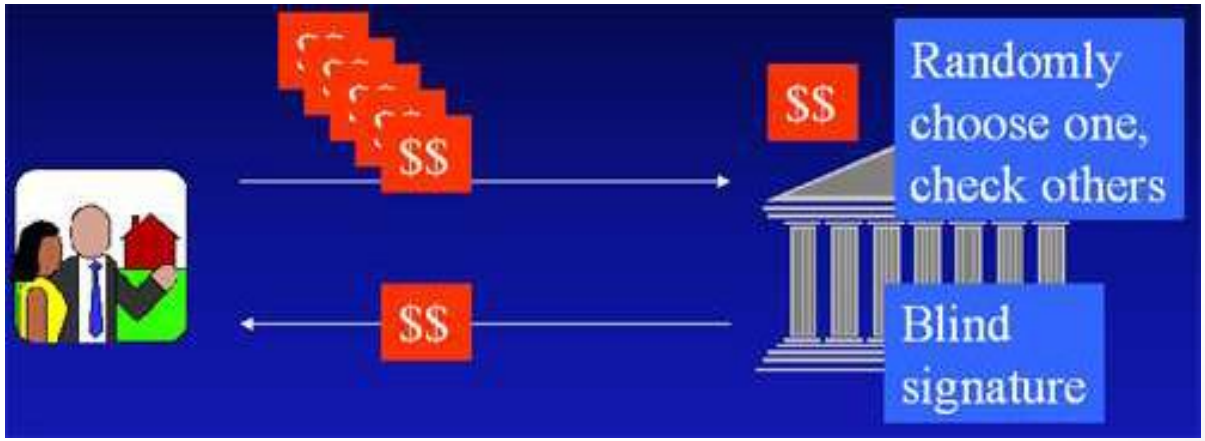
- Обналичивание - новый вариант
  - 1) Участник просит Банк выдать 100$
  - 2) Участник готовит счет: { Я счет на 100$ #4257 }
  - 3) Банк вслепую подписывает счет
  - 4) Участник проверяет подпись и признает счет

- Остались проблемы:
  - 1) Двойной траты
  - 2) Банк подпишет 1000$ вместо 100$

Проверка выдаваемой суммы

- Выборочная проверка
  - 1) Участник посылает Банку 1000 счетов
  - 2) Банк проверяет (открывая) 999 и подписывает 1000-ый счет

- Система ключей:
  - 1) У банка есть набор секретных ключей SK_1, SK_{10}, SK_{100}, . . .
  - 2) Каждая подпись годится только для фиксированной суммы



Контроль двойной траты

- On-line контроль
  - Продавец высылает запрос Банку

    “Была ли уже потрачена купюра 4257?”
- Off-line контроль:
  - 1) Участник генерирует $x_1, \ldots x_k, y_1, \ldots, y_k$ так, что $ID = x_i \oplus y_i$
  - 2) Участник посылает Банку хэш-функции от этих значений
  - 3) К каждой электронной купюре добавляется набор из k значений, по одному из пары по выбору Продавца
  - 4) Два раза потратили $\Rightarrow$ с вероятностью $1 - 2^{-k}$ можно установить нарушителя

Пошаговая схема протокола