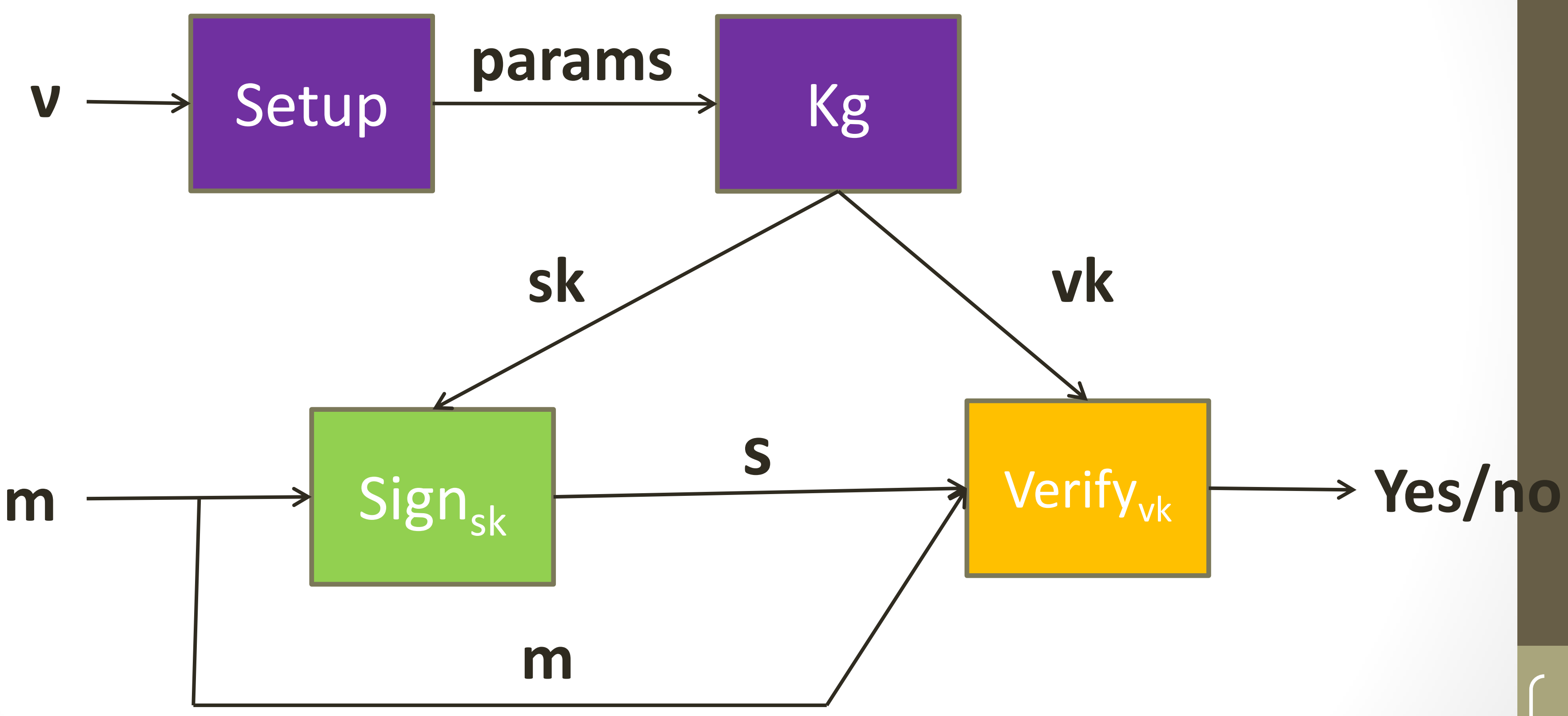
КРИПТОГРАФИЧЕСКАЯ РЕАЛИЗАЦИЯ

Цифровая подпись



Неподделываемость при CCA (UF-CCA)

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

π

Public Key

vk

m_i

s_i

par ← Setup(n)

(vk,sk) ← Kg (par)

$s_i \leftarrow \mathrm{Sign}_{sk}(m_i)$

win ← Verify(vk,m*,s*)

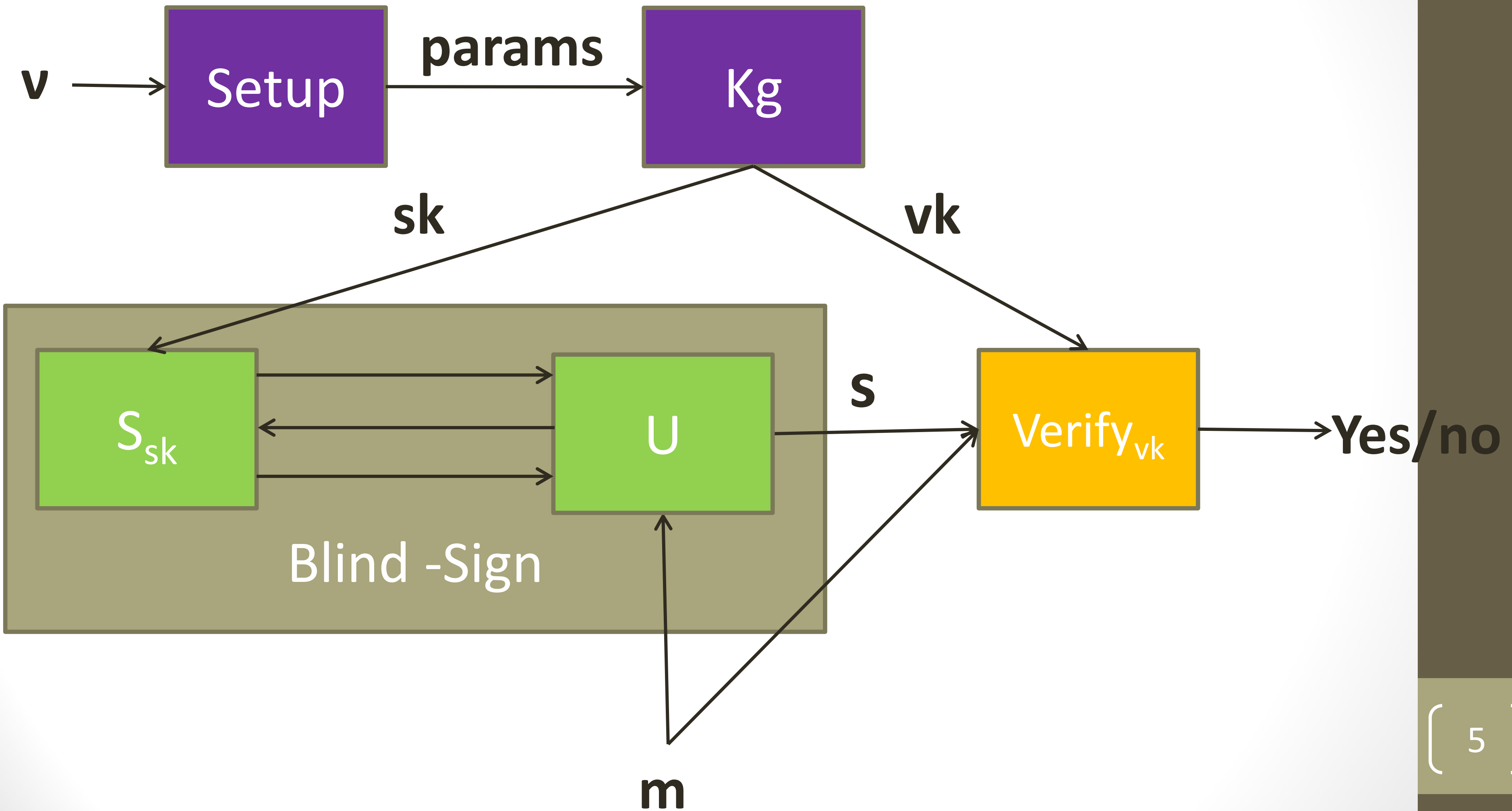
UF-CCA security: ∀ pol атакующего  ∃ neg функция f(x)
Prob [win] ≤ f(n)

win

Хэш функция с полной областью определения

- **Определение:**
  - **Keygen(v)**: генерация модуля RSA $N=PQ$, пары d и e: $ed=1 \bmod \Phi(N)$. 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 $H(m)^d \bmod N$
  - **Verify((N,e),m,s)**:проверка $s^e= H(m) \bmod N$
- **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)

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

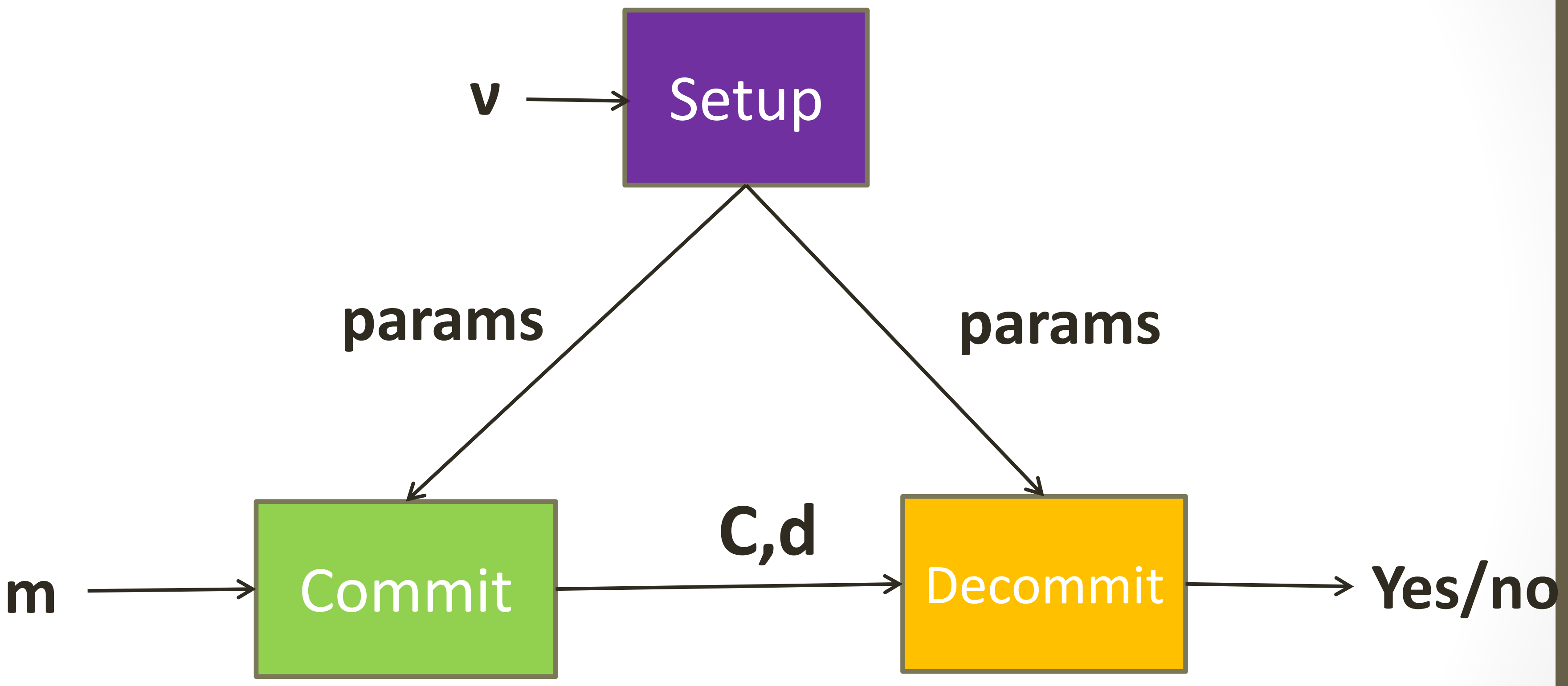
$$s = t/r = H(m)^d \bmod n$$

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 <u>вычислительно стойкое</u>; если в схеме доказательства получатель вычислительно не ограничен, то сокрытие <u>совершенно стойкое</u>
- **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 $C=g^x h^r$ ($C=g^x(g^a)^r=g^{x+ar}$)

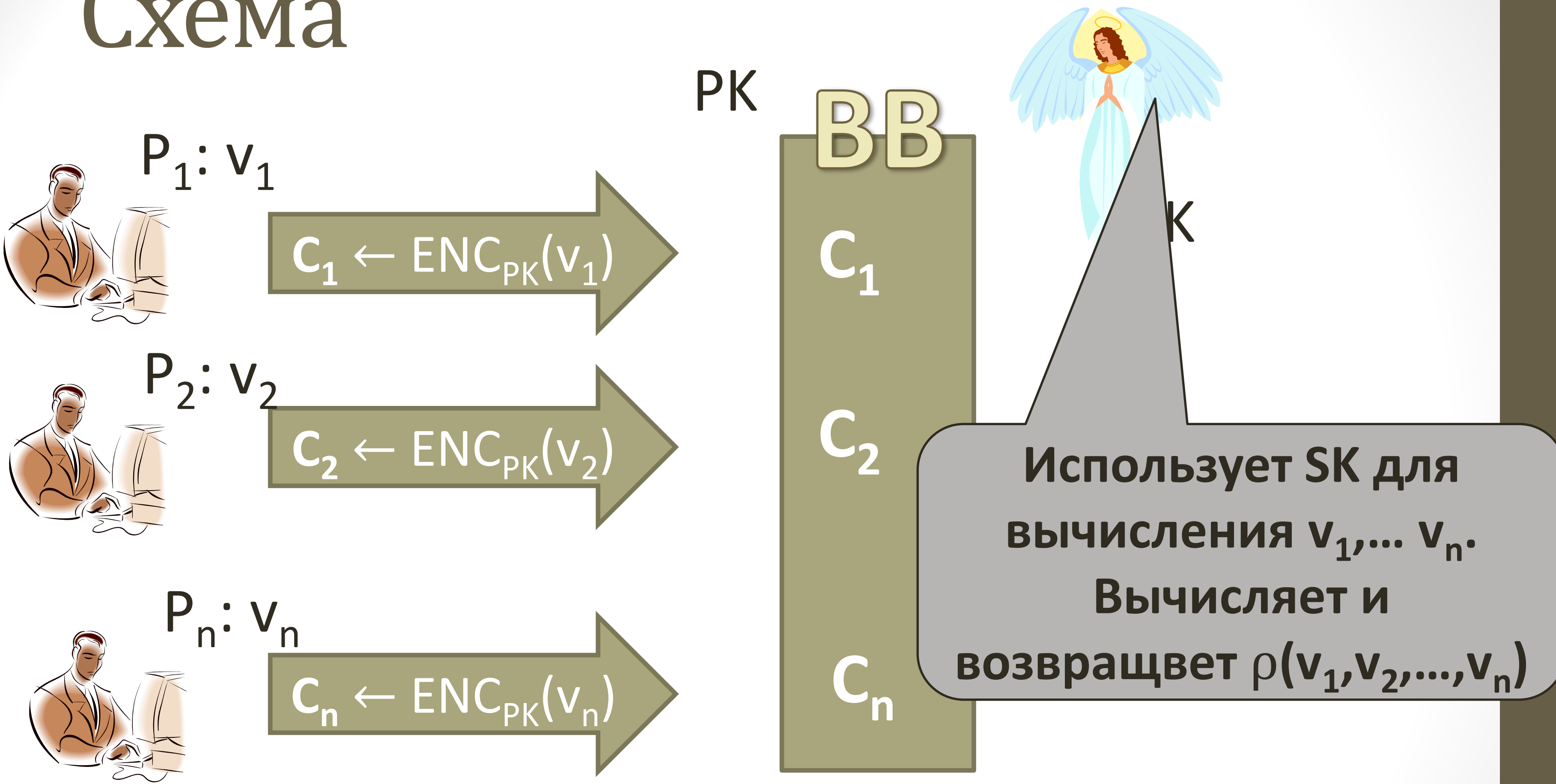
- **Decommit(C,x,r):** проверка $C=g^x h^r$

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



PK
BB
P1: v1
C1 ← ENCPK(v1)
P2: v2
C2 ← ENCPK(v2)
Pn: vn
Cn ← ENCPK(vn)
C1
C2
Cn
K
Использует SK для вычисления v1,… vn. Вычисляет и возвращвет ρ(v1,v2,…,vn)

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

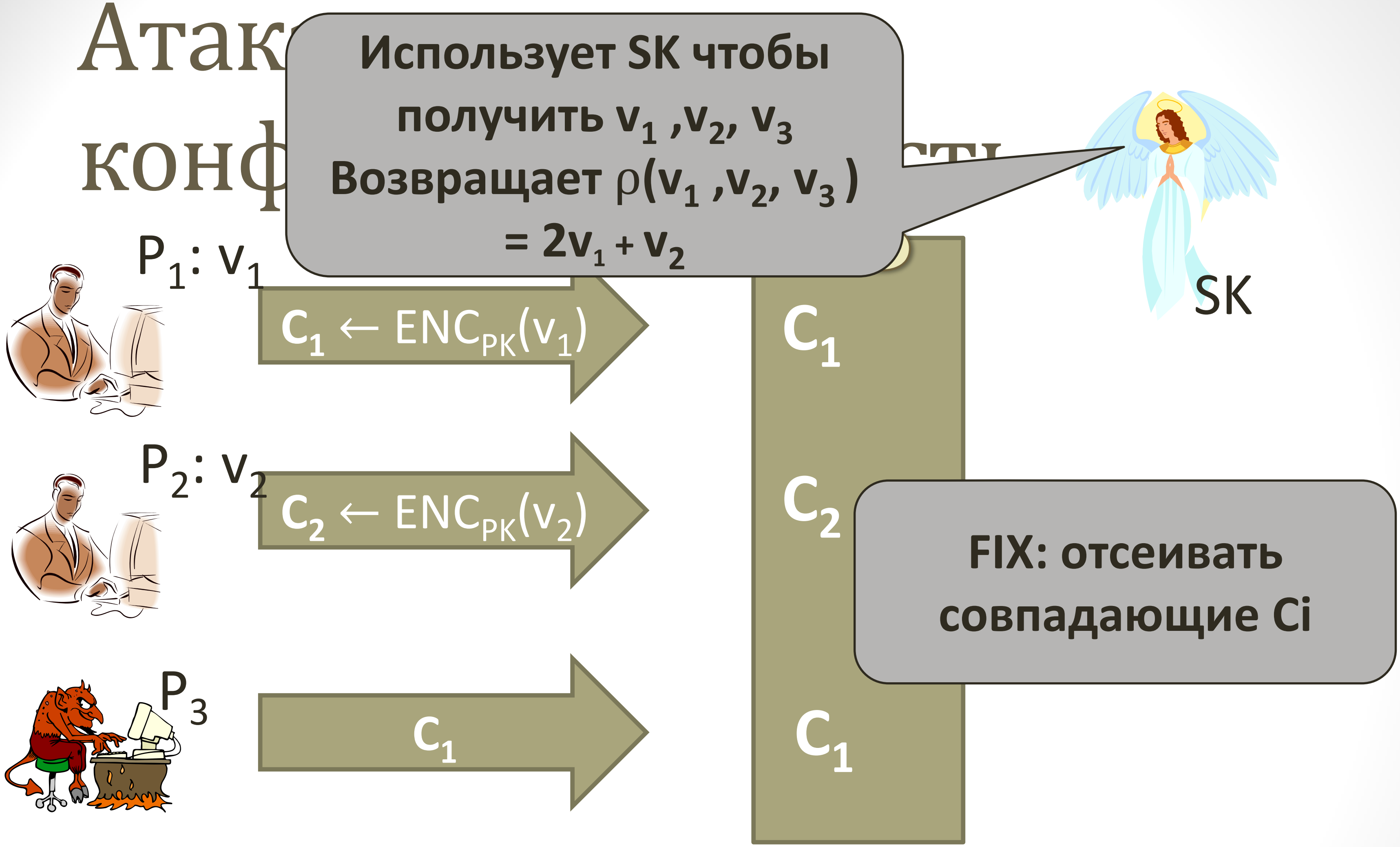
- Пусть π=(KG,ENC,DEC) гомоморфная схема шифрования. Enc2Vote(π) :
- **Setup(v)**: KG генерирует (SK,PK,[])
- **Vote(PK,v)**: b $\leftarrow$ ENC_{PK}(v)
- **Process Ballot([BB],b)**: [BB] $\leftarrow$ [BB,b]
- **Tallying([BB],x)**: где [BB] = [b_1,b_2,…,b_n]

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

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

  возвращает **result**

Атак конф



- 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 ← $\text{Enc}_{PK}(M_b)$

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

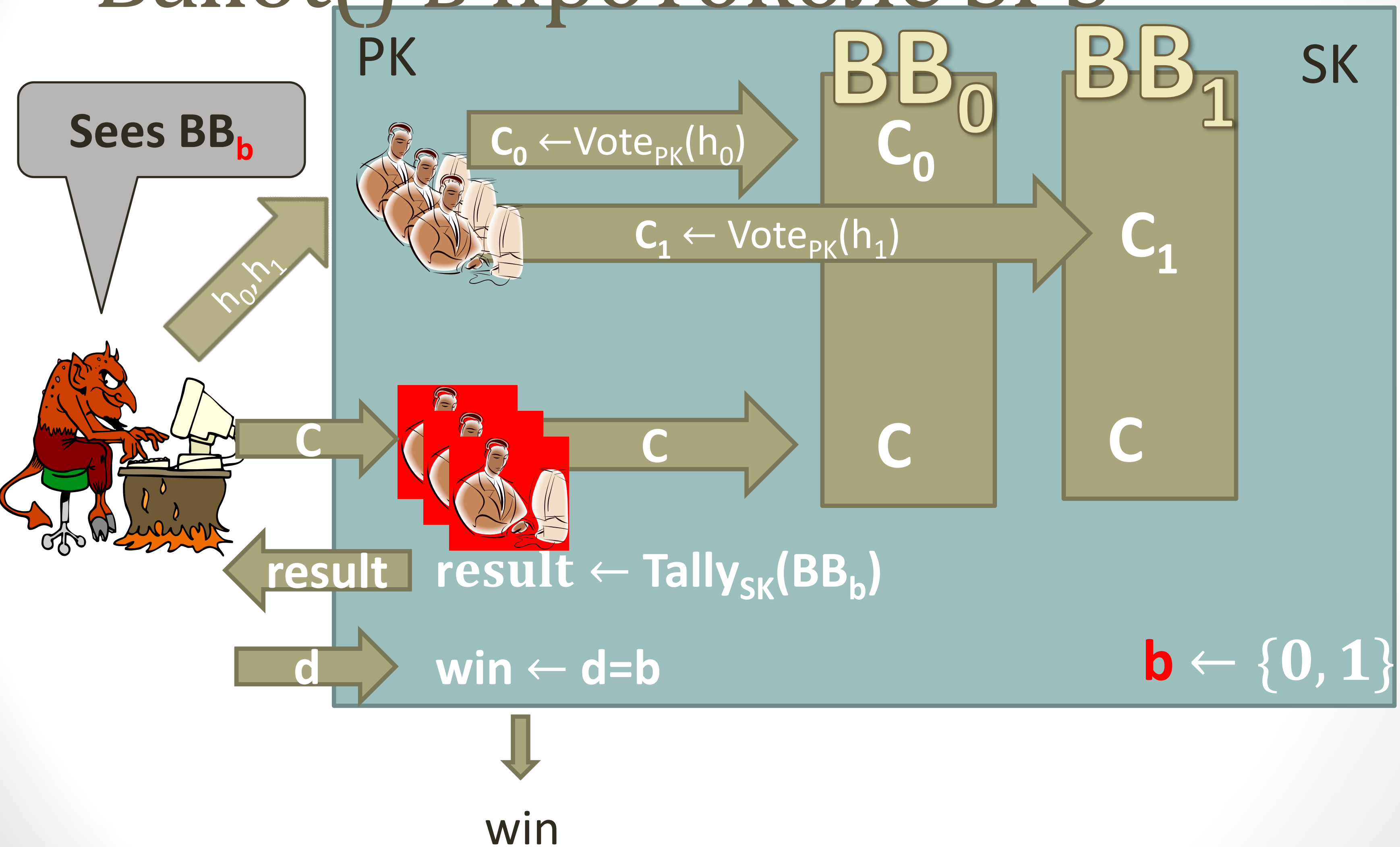
win ← d=b

win

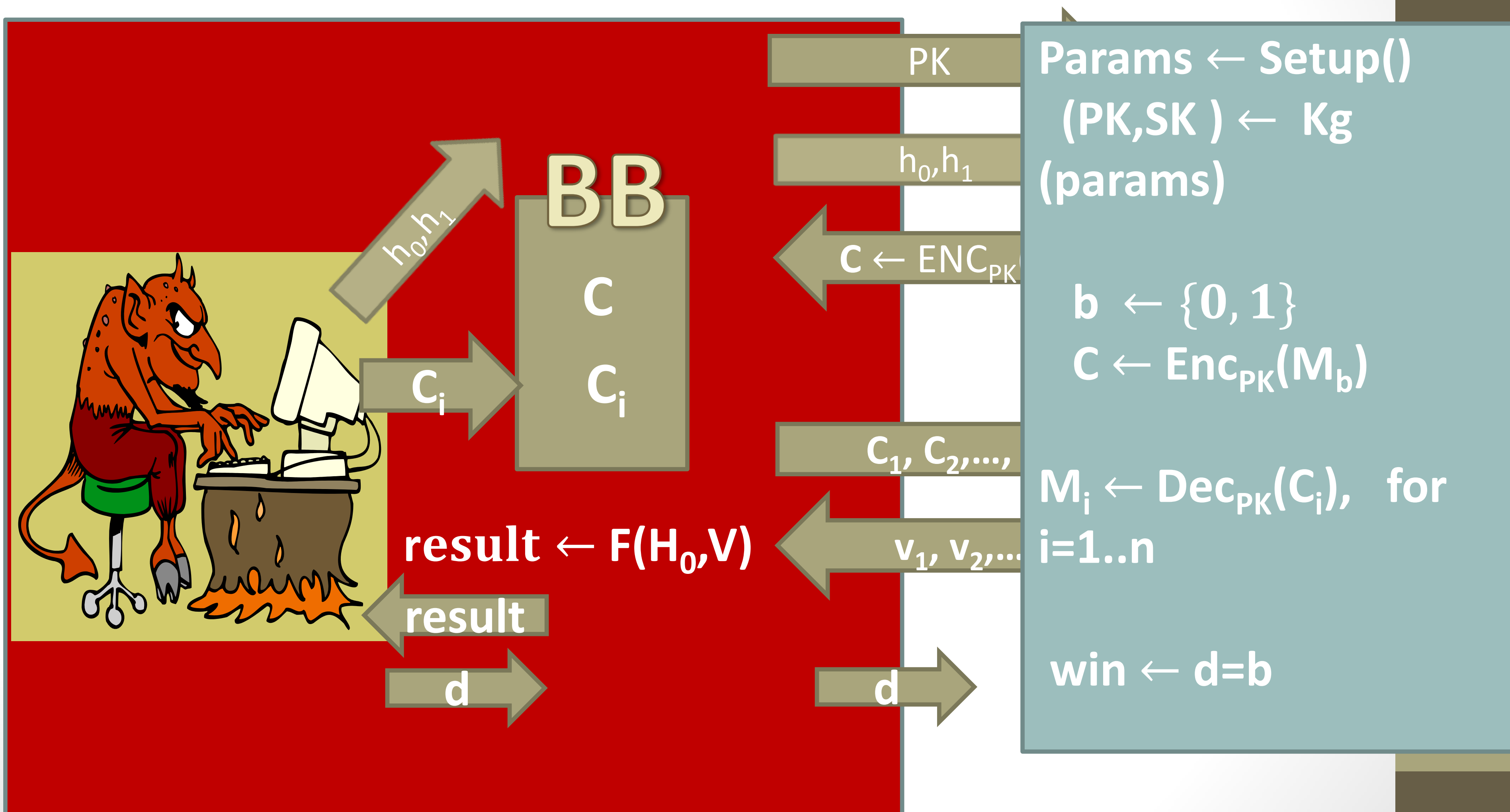
ElGamal is not non-malleable

- 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

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

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