МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

Протокол анонимности Нурми-Саломаа-Сантина

ОТЧЁТ
ПО ДИСЦИПЛИНЕ
«КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ»

студента 5 курса 531 группы
специальности 10.05.01 Компьютерная безопасность
факультета компьютерных наук и информационных технологий
Норикова Павла Сергеевича

Преподаватель
аспирант _____ Р. А. Фарахутдинов
подпись, дата

Саратов 2023

ВВЕДЕНИЕ

Цель работы – изучение и реализация протокола анонимности Нурми-Саломаа-Сантина.

1 Теория

Протоколы электронного голосования – это протоколы обмена данными для безопасного голосования через электронные технические средства.

Многие страны уже внедряют данный тип голосования и для уверенности в целостности, конфиденциальности и доступности выборов используют протоколы с доказанной защищенностью, которые должны реализовывать обязательные требования в том, что только голосующий может знать свой выбор, что проголосовать можно один раз и только участником, допущенным к выборам и решение проголосовавшего не может быть кем-либо изменено.

1.1 Описание алгоритма

Обозначения: V – валидатор, B – избиратель, A – агентство, M – секретные опознавательные метки.

1. V отправляет M всем B до голосования.
2. V отправляет A весь набор M, но без информации о том, кому они принадлежат.
3. B создает свои ключи $K_{B_{\text{зак}}}$, $K_{B_{\text{отк}}}$ и выкладывает в общий доступ $K_{B_{\text{отк}}}$, а также создает секретный ключ ($K_{B_{\text{сек}}}$), который нужен, чтобы никто не узнал содержимое бюллетеня до нужного момента.
4. B формирует сообщение C, где выражает свой выбор, подписывает $K_{B_{\text{зак}}}$, прикладывает к нему полученную M и шифрует $K_{B_{\text{сек}}}$.
5. К зашифрованному тексту B прикладывает M и отправляет A.
6. A получает зашифрованный текст, по M определяет, что он пришел от B, но не знает от кого именно и как B проголосовал, после публикует его.
7. Опубликованный зашифрованный текст служит информацией, чтобы B отправил $K_{B_{\text{сек}}}$.

8. A собирает ключи, расшифровывает текст, подсчитывает голоса и присоединяет к опубликованному зашифрованному тексту C без M.

2 Практическая реализация

2.1 Описание программы

Функции *encr* и *decr* осуществляют шифрование и расшифрование соответственно.

Функции *gen_p*, *gen_q* и *gen_g* генерируют числа *p*, *q* и *g* соответственно.

Функции *gcd_ex* и *gcd* реализуют расширенный и обычный алгоритм Евклида соответственно.

Функция *V1* генерирует опознавательные метки.

Класс *B* включает в себя все необходимые данные и ключи избирателя, а также метод *B1*, реализующий процесс голосования.

На вход программе подается количество избирателей и кандидатов.

2.2 Тестирование программы

```
Введите количество избирателей: 10
Введите количество кандидатов: 4

1. V отправляет M всем B до голосования
Избиратель 0: b'9OzlaxNsL-demC-5v6C22nmXm3efWZLfS_H5-YxwJwI='
Избиратель 1: b'SA2yPbGSLtWRvO8y095lCh6d2hWxEOo_TjhDCnKN6X4='
Избиратель 2: b'6YQSwc2EUQkhcuUKbcAFpajRQ26pdRTv6Ao20K0yDuI='
Избиратель 3: b'25PAc81_uNjTBkqnVG2NgkVQ2AIbnTyWB2hS8pUGXT0='
Избиратель 4: b'zhWw0mLfso4G7D7cm6o8gxfZ0DdssgCMDb_yHc1Ians='
Избиратель 5: b'crRKfaLg6pQP-YKmjbRYCod2GBcbiwhPG6UsukG088k='
Избиратель 6: b'eZ55xovnmX7ND8KSy5rl7rXJ2RbRwYlte56B5mg5tpY='
Избиратель 7: b'xsqTYxgp_Ui4GhpPVm6UfWxon-XeuMDDva8a3NA9r9M='
Избиратель 8: b'_TP8vtD1_szSM76cd4zqO6JJwvCRQYWtKOq_IfqykoU='
Избиратель 9: b'nWQtXZtlgIOP2Ia2Nf1uhyx3Witw8NkiOvffj_vHOp4='

2. V отправляет A весь набор M, но без информации о том, кому они принадлежат.

3. B создает свои ключи K_зак , K_отк и выкладывает в общий доступ K_отк , а также создает секретный ключ (K_сек ), который нужен, чтобы никто не узнал содержимое бюллетеня до нужного момента.
K_зак и K_отк будут одинаковыми для всех пользователей
4510024835505916026 45225780786689089339728276598558187813612884101

K_сек:
Избиратель 1: b'SpQumHYQ0dhqHwSpmP_Emlmetp96GBDWUPM7-Qg10JQ='
Избиратель 2: b'FHyiysavyUjOP9HMXzw2ewomNbzEjfFn5-sIhRKoUr8='
Избиратель 3: b'U57EeZ6w96S8Di37WTxNBn6DUKRxbd_JRXOXGcFsX00='
Избиратель 4: b'AuE1ukcEogyXqB9jZCvq4kFz45CJ2LRjT_s4xSf6M18='
Избиратель 5: b'2Wab4Qk7lB_ngJY2uTwlpA31Fvx1UlhjF67heiL3hPw='
Избиратель 6: b'X5T4LOZCCG037KpSeDwZj7Cya7wWKTqRkh9-znw8dLw='
Избиратель 7: b'pOpW-jgPXeVc_9YoFBge1_n4sW2QgIzlAEZ6BRsznm8='
Избиратель 8: b'U22_bXmLGhQLneB4STv9q6nXTzf24b0b40w_-5-tekA='
Избиратель 9: b'7pUNMn1zg5xm8YuOKOLfQppcGjfaBviOFELBXI3hgTE='
Избиратель 10: b'IlNmm5OMIPbkcBrkGnW-M8JzIWBtRRDZylam9x6o5wg='
```

Рисунок 1 – Шаги 1-3

```
4. В формирует сообщение C, где выражает свой выбор, подписывает K_зак , прикладывает к нему полученную M и шифрует K_сек.
Избиратель 1:
Выбор: 2
Подпись: (2, 413476397356752059, 5069983092854622176)
Прикладываем M и шифруем: b'gAAAAABljDe22L5Sz2AJ5F5GuMzz-l55ss8T4n6GfjjWyfJYGHN4RYt4s2IBSpubszOOcPy_3XMoSQMH2cAxTRXQsZ8Zaxmt
7HkzwK1AYnyvQLKfs5yBBcoYwmHb28z7deOR2fXcGwupYi1JULKKftqib6xJnpTF2uGbcyxT5-VAuO9nkeW7nEDXnqcZoKBalmDkfuUBaeho'
Избиратель 2:
Выбор: 2
Подпись: (2, 1108779033457629734, 5152321561422802820)
Прикладываем M и шифруем: b'gAAAAABljDe22JT4dn3rNyjjPOdvsD2axQaL2N-OnqIkJBykNOE9Rxd4EdRKBaGE_8pdm2GpZSxO-yFK4E71rlCsYWfgvrMI
kpgA5f7voJmJao3QclA5Ct3vU7T3MZ7e_FHdTfrTdvP1VgNCEgOvbXF8tq8TpkfuJpMUEbBLzhVq2jVH7Cby44SCWkZcJ6iXc2zg4vga_Rq3'
Избиратель 3:
Выбор: 4
Подпись: (4, 2094113262406277523, 4806558376291697362)
Прикладываем M и шифруем: b'gAAAAABljDe2_-INnF4kgbl745Mnlf2LQ-ZhX3KtMI0473CDGvdhSlCQR1_VM7kTH9mcTUVARn_NuSJCH-6uSX5UsA297eFB
5V6ZQKbWJYXAua_4xeFyIyxMs9TixPEJ4kDYxlnXkcYhOKZOBN861I0Z2p5WXRi4Dt8Q3B7uu2_KpeQYFDcXEcNjKDg7c3XZfXIQcUrHXg8T'
Избиратель 4:
Выбор: 3
Подпись: (3, 2151251717267786912, 251203007157763611)
Прикладываем M и шифруем: b'gAAAAABljDe2msryZ8lu5g5Bcwr30Bv2zkDBMPxcVnuhLfyJxaC5cJJ6so8sxWsysdRPJEDvK6hKP9lNOCsQP5z-TtlRdByi
B8qSUqPuS46Cl1t9Rtg3ecoRwxWDhPI_XDFP7ypSYGxqcnIKLrXwrHOPXuTWbpy1rA2HVfilsstZJ4rb-dJ3_ImcOhoWEaj8VIR1A0iErCVZ'
Избиратель 5:
Выбор: 2
Подпись: (2, 1863362303289445347, 3997940255599243418)
Прикладываем M и шифруем: b'gAAAAABljDe2xaY895tdVBLrcNKYd_4WZ7IJTqN50bi4chniErclm9a1_fb308PAKHxG7qEPvwGPI7LIUrB8pKw_Fc0Wtipm
5sL6yYNFPkZlYPerZqGcZFJ-l_oskfyMXCOSqnZVL2Jb4oU9ThTqidYZyrxNyCpQsXkoELWLfLU68PItZNI61E6zSk0Cj3jqC5hgOd11y4LF'
Избиратель 6:
Выбор: 3
Подпись: (3, 2919558590681882247, 4306592027122953323)
Прикладываем M и шифруем: b'gAAAAABljDe2PVH8OqhsYQPOIGBzp20cxaO9emEYVmrMQ4W7sMFogtMaM1kmBYNu8vwCc6KU2nDPOxynqio0xG5MRzs7lDS7
tyA_ETHDMqsWAPeqrTQWeVy88O1R5gr62eLzEwBnxiWso19D4-yVEbJ0N8Wru1_AehSbRx10GqWGJWIQqhNo_E80myAcY8dP0NNJ9aiC16mD'
Избиратель 7:
Выбор: 2
Подпись: (2, 2515240726527054903, 2671265931672505297)
Прикладываем M и шифруем: b'gAAAAABljDe2y80UvLwwIqgX60Bbw8T6AeKEoSTKRzkalvJ2iCtn2YHskxXpqftwyBIAXePl5yWNMnRgunwK3_igCM6Dgvk0
t52LbITROFn8Q2PzHbIKys9VZ36f0bR_K1HLpCJbqYJZHT7r-o4IPfInTSauxKjWsYlrRk0F7aWwCNaaq0p_MzDJaLVZVrhCUmljnFtiNAPS'
Избиратель 8:
Выбор: 2
Подпись: (2, 3815080868483731516, 2427066541923074281)
Прикладываем M и шифруем: b'gAAAAABljDe2s3zP507sPeXdAL_-9MWunkK3-M9WRUzHJNd4WZISuJ4GsFJxYp7zeaqqKOHbgAStH0QBfRnqYN0JeIfmQsL-
ZLSXNIBUxK7Oe5jBJHiwzg3pwvHdwthuQdOcJDe8cMh7C6lO-WCxoZFN_axFbf7c5w5FMiC28t1L0K8mzr3Da0zTBnm3JmdYUPEURJdvD85m'
Избиратель 9:
Выбор: 4
Подпись: (4, 3236003333635824830, 3334629252848340207)
Прикладываем M и шифруем: b'gAAAAABljDe2ixifBb_q6_bx3qBZfy-CRzIH-f-JVk3uuK_v0wPN3-2Kzw4w2hfJ6hZ-gMDw8HpbiCNpYhMe-TLkQvhZCjn4
O_UitqJTja9cAJ4MYSlxDLqCRrM2kz4kFtJR5UOr0kjvhy-xTk_HYDVmCZaPo01N-izYz_7eagkIlh-n_eMZcMct-Oz6NSVkfR3D1ik6OW4Q'
Избиратель 10:
Выбор: 3
Подпись: (3, 3167085699569798295, 3714208851795717153)
Прикладываем M и шифруем: b'gAAAAABljDe2vVDc-PUyVmsAqwKKfaDjvfxJkdzW4XnOlVFBh7h8XkKRBEePFzn4NCezcg63j8uO2REpWvpKZYvt7fG2rbU5
05_Pcpg-eGnYifO9xNLS46_BZzp_cD-vrU_b7IMoY57RWO8jrbY85KSAqNw0-nwhTr47HR2T7Hi9_hEGHMCkzXh1R1hFn9T77TMngZMTvgtL'
```

Рисунок 2 – Шаг 4

Рисунок 3 – Шаг 5

Рисунок 4 – Шаг 6

```
7. Опубликованный зашифрованный текст служит информацией, чтобы B отправил K_сек.
Для сообщения (b'gAAAAABljDe22L5Sz2AJ5F5GuMzz-l55ss8T4n6GfjjWyfJYGHN4RYt4s2IBSpubszOOcPy_3XMoSQMH2cAxTRXQsZ8Zaxmt7HkzwK1AYnyvQLKfs5yBBcoYwmHb28z7deOR2fXcGwupYi1JULKKftqib6xJnpTF2uGbcyxT5-VAuO9nkeW7nEDXnqcZoKBalmDkfuUBaeho', b'9OzlaxNsL-demC-5v6C22nmXm3efWZLfS_H5-YxwJwI=') ключ - b'SpQumHYQ0dhqHwSpmP_Emlmetp96GBDWUPM7-Qg10JQ='
Для сообщения (b'gAAAAABljDe22JT4dn3rNyjjPOdvsD2axQaL2N-OnqIkJBykNOE9Rxd4EdRKBaGE_8pdm2GpZSxO-yFK4E71rlCsYWfgvrMIkpgA5f7voJmJao3QclA5Ct3vU7T3MZ7e_FHdTfrTdvP1VgNCEgOvbXF8tq8TpkfuJpMUEbBLzhVq2jVH7Cby44SCWkZcJ6iXc2zg4vga_Rq3', b'SA2yPbGSLtWRvO8y095lCh6d2hWxEOo_TjhDCnKN6X4=') ключ - b'FHyiysavyUjOP9HMXzw2ewomNbzEjfFn5-sIhRKoUr8='
Для сообщения (b'gAAAAABljDe2_-INnF4kgbl745Mnlf2LQ-ZhX3KtMI0473CDGvdhSlCQR1_VM7kTH9mcTUVARn_NuSJCH-6uSX5UsA297eFB5V6ZQKbWJYXAua_4xeFyIyxMs9TixPEJ4kDYxlnXkcYhOKZOBN861I0Z2p5WXRi4Dt8Q3B7uu2_KpeQYFDcXEcNjKDg7c3XZfXIQcUrHXg8T', b'6YQSwc2EUQkhcuUKbcAFpajRQ26pdRTv6Ao20K0yDuI=') ключ - b'U57EeZ6w96S8Di37WTxNBn6DUKRxbd_JRXOXGcFsX00='
Для сообщения (b'gAAAAABljDe2msryZ8lu5g5Bcwr30Bv2zkDBMPxcVnuhLfyJxaC5cJJ6so8sxWsysdRPJEDvK6hKP9lNOCsQP5z-TtlRdByiB8qSUqPuS46Cl1t9Rtg3ecoRwxWDhPI_XDFP7ypSYGxqcnIKLrXwrHOPXuTWbpy1rA2HVfilsstZJ4rb-dJ3_ImcOhoWEaj8VIR1A0iErCVZ', b'25PAc81_uNjTBkqnVG2NgkVQ2AIbnTyWB2hS8pUGXT0=') ключ - b'AuE1ukcEogyXqB9jZCvq4kFz45CJ2LRjT_s4xSf6M18='
Для сообщения (b'gAAAAABljDe2xaY895tdVBLrcNKYd_4WZ7IJTqN50bi4chniErclm9a1_fb308PAKHxG7qEPvwGPI7LIUrB8pKw_Fc0Wtipm5sL6yYNFPkZlYPerZqGcZFJ-l_oskfyMXCOSqnZVL2Jb4oU9ThTqidYZyrxNyCpQsXkoELWLfLU68PItZNI61E6zSk0Cj3jqC5hgOd11y4LF', b'zhWw0mLfso4G7D7cm6o8gxfZ0DdssgCMDb_yHc1Ians=') ключ - b'2Wab4Qk7lB_ngJY2uTwlpA31Fvx1UlhjF67heiL3hPw='
Для сообщения (b'gAAAAABljDe2PVH8OqhsYQPOIGBzp20cxaO9emEYVmrMQ4W7sMFogtMaM1kmBYNu8vwCc6KU2nDPOxynqio0xG5MRzs7lDS7tyA_ETHDMqsWAPeqrTQWeVy88O1R5gr62eLzEwBnxiWso19D4-yVEbJ0N8Wru1_AehSbRx10GqWGJWIQqhNo_E80myAcY8dP0NNJ9aiC16mD', b'crRKfaLg6pQP-YKmjbRYCod2GBcbiwhPG6UsukG088k=') ключ - b'X5T4LOZCCG037KpSeDwZj7Cya7wWKTqRkh9-znw8dLw='
Для сообщения (b'gAAAAABljDe2y80UvLwwIqgX60Bbw8T6AeKEoSTKRzkalvJ2iCtn2YHskxXpqftwyBIAXePl5yWNMnRgunwK3_igCM6Dgvk0t52LbITROFn8Q2PzHbIKys9VZ36f0bR_K1HLpCJbqYJZHT7r-o4IPfInTSauxKjWsYlrRk0F7aWwCNaaq0p_MzDJaLVZVrhCUmljnFtiNAPS', b'eZ55xovnmX7ND8KSy5rl7rXJ2RbRwYlte56B5mg5tpY=') ключ - b'pOpW-jgPXeVc_9YoFBge1_n4sW2QgIzlAEZ6BRsznm8='
Для сообщения (b'gAAAAABljDe2s3zP507sPeXdAL_-9MWunkK3-M9WRUzHJNd4WZISuJ4GsFJxYp7zeaqqKOHbgAStH0QBfRnqYN0JeIfmQsL-ZLSXNIBUxK7Oe5jBJHiwzg3pwvHdwthuQdOcJDe8cMh7C6lO-WCxoZFN_axFbf7c5w5FMiC28t1L0K8mzr3Da0zTBnm3JmdYUPEURJdvD85m', b'xsqTYxgp_Ui4GhpPVm6UfWxon-XeuMDDva8a3NA9r9M=') ключ - b'U22_bXmLGhQLneB4STv9q6nXTzf24b0b40w_-5-tekA='
Для сообщения (b'gAAAAABljDe2ixifBb_q6_bx3qBZfy-CRzIH-f-JVk3uuK_v0wPN3-2Kzw4w2hfJ6hZ-gMDw8HpbiCNpYhMe-TLkQvhZCjn4O_UitqJTja9cAJ4MYSlxDLqCRrM2kz4kFtJR5UOr0kjvhy-xTk_HYDVmCZaPo01N-izYz_7eagkIlh-n_eMZcMct-Oz6NSVkfR3D1ik6OW4Q', b'_TP8vtD1_szSM76cd4zqO6JJwvCRQYWtKOq_IfqykoU=') ключ - b'7pUNMn1zg5xm8YuOKOLfQppcGjfaBviOFELBXI3hgTE='
Для сообщения (b'gAAAAABljDe2vVDc-PUyVmsAqwKKfaDjvfxJkdzW4XnOlVFBh7h8XkKRBEePFzn4NCezcg63j8uO2REpWvpKZYvt7fG2rbU505_Pcpg-eGnYifO9xNLS46_BZzp_cD-vrU_b7IMoY57RWO8jrbY85KSAqNw0-nwhTr47HR2T7Hi9_hEGHMCkzXh1R1hFn9T77TMngZMTvgtL', b'nWQtXZtlgIOP2Ia2Nf1uhyx3Witw8NkiOvffj_vHOp4=') ключ - b'IlNmm5OMIPbkcBrkGnW-M8JzIWBtRRDZylam9x6o5wg='
```

Рисунок 5 – Шаг 7

```
8. A собирает ключи, расшифровывает текст, подсчитывает голоса и присоединяет к опубликованному зашифрованному тексту C без M
2 (b'gAAAAABljDe22L5Sz2AJ5F5GuMzz-l55ss8T4n6GfjjWyfJYGHN4RYt4s2IBSpubszOOcPy_3XMoSQMH2cAxTRXQsZ8Zaxmt7HkzwK1AYnyvQLKfs5yBBcoYwmHb28z7deOR2fXcGwupYi1JULKKftqib6xJnpTF2uGbcyxT5-VAuO9nkeW7nEDXnqcZoKBalmDkfuUBaeho', b'9OzlaxNsL-demC-5v6C22nmXm3efWZLfS_H5-YxwJwI=')
2 (b'gAAAAABljDe22L5Sz2AJ5F5GuMzz-l55ss8T4n6GfjjWyfJYGHN4RYt4s2IBSpubszOOcPy_3XMoSQMH2cAxTRXQsZ8Zaxmt7HkzwK1AYnyvQLKfs5yBBcoYwmHb28z7deOR2fXcGwupYi1JULKKftqib6xJnpTF2uGbcyxT5-VAuO9nkeW7nEDXnqcZoKBalmDkfuUBaeho', b'9OzlaxNsL-demC-5v6C22nmXm3efWZLfS_H5-YxwJwI=')
4 (b'gAAAAABljDe22L5Sz2AJ5F5GuMzz-l55ss8T4n6GfjjWyfJYGHN4RYt4s2IBSpubszOOcPy_3XMoSQMH2cAxTRXQsZ8Zaxmt7HkzwK1AYnyvQLKfs5yBBcoYwmHb28z7deOR2fXcGwupYi1JULKKftqib6xJnpTF2uGbcyxT5-VAuO9nkeW7nEDXnqcZoKBalmDkfuUBaeho', b'9OzlaxNsL-demC-5v6C22nmXm3efWZLfS_H5-YxwJwI=')
3 (b'gAAAAABljDe22L5Sz2AJ5F5GuMzz-l55ss8T4n6GfjjWyfJYGHN4RYt4s2IBSpubszOOcPy_3XMoSQMH2cAxTRXQsZ8Zaxmt7HkzwK1AYnyvQLKfs5yBBcoYwmHb28z7deOR2fXcGwupYi1JULKKftqib6xJnpTF2uGbcyxT5-VAuO9nkeW7nEDXnqcZoKBalmDkfuUBaeho', b'9OzlaxNsL-demC-5v6C22nmXm3efWZLfS_H5-YxwJwI=')
2 (b'gAAAAABljDe22L5Sz2AJ5F5GuMzz-l55ss8T4n6GfjjWyfJYGHN4RYt4s2IBSpubszOOcPy_3XMoSQMH2cAxTRXQsZ8Zaxmt7HkzwK1AYnyvQLKfs5yBBcoYwmHb28z7deOR2fXcGwupYi1JULKKftqib6xJnpTF2uGbcyxT5-VAuO9nkeW7nEDXnqcZoKBalmDkfuUBaeho', b'9OzlaxNsL-demC-5v6C22nmXm3efWZLfS_H5-YxwJwI=')
3 (b'gAAAAABljDe22L5Sz2AJ5F5GuMzz-l55ss8T4n6GfjjWyfJYGHN4RYt4s2IBSpubszOOcPy_3XMoSQMH2cAxTRXQsZ8Zaxmt7HkzwK1AYnyvQLKfs5yBBcoYwmHb28z7deOR2fXcGwupYi1JULKKftqib6xJnpTF2uGbcyxT5-VAuO9nkeW7nEDXnqcZoKBalmDkfuUBaeho', b'9OzlaxNsL-demC-5v6C22nmXm3efWZLfS_H5-YxwJwI=')
2 (b'gAAAAABljDe22L5Sz2AJ5F5GuMzz-l55ss8T4n6GfjjWyfJYGHN4RYt4s2IBSpubszOOcPy_3XMoSQMH2cAxTRXQsZ8Zaxmt7HkzwK1AYnyvQLKfs5yBBcoYwmHb28z7deOR2fXcGwupYi1JULKKftqib6xJnpTF2uGbcyxT5-VAuO9nkeW7nEDXnqcZoKBalmDkfuUBaeho', b'9OzlaxNsL-demC-5v6C22nmXm3efWZLfS_H5-YxwJwI=')
2 (b'gAAAAABljDe22L5Sz2AJ5F5GuMzz-l55ss8T4n6GfjjWyfJYGHN4RYt4s2IBSpubszOOcPy_3XMoSQMH2cAxTRXQsZ8Zaxmt7HkzwK1AYnyvQLKfs5yBBcoYwmHb28z7deOR2fXcGwupYi1JULKKftqib6xJnpTF2uGbcyxT5-VAuO9nkeW7nEDXnqcZoKBalmDkfuUBaeho', b'9OzlaxNsL-demC-5v6C22nmXm3efWZLfS_H5-YxwJwI=')
4 (b'gAAAAABljDe22L5Sz2AJ5F5GuMzz-l55ss8T4n6GfjjWyfJYGHN4RYt4s2IBSpubszOOcPy_3XMoSQMH2cAxTRXQsZ8Zaxmt7HkzwK1AYnyvQLKfs5yBBcoYwmHb28z7deOR2fXcGwupYi1JULKKftqib6xJnpTF2uGbcyxT5-VAuO9nkeW7nEDXnqcZoKBalmDkfuUBaeho', b'9OzlaxNsL-demC-5v6C22nmXm3efWZLfS_H5-YxwJwI=')
3 (b'gAAAAABljDe22L5Sz2AJ5F5GuMzz-l55ss8T4n6GfjjWyfJYGHN4RYt4s2IBSpubszOOcPy_3XMoSQMH2cAxTRXQsZ8Zaxmt7HkzwK1AYnyvQLKfs5yBBcoYwmHb28z7deOR2fXcGwupYi1JULKKftqib6xJnpTF2uGbcyxT5-VAuO9nkeW7nEDXnqcZoKBalmDkfuUBaeho', b'9OzlaxNsL-demC-5v6C22nmXm3efWZLfS_H5-YxwJwI=')
Результаты голосования:
Кандидат 1: 0
Кандидат 2: 5
Кандидат 3: 3
Кандидат 4: 2
```

Рисунок 5 – Шаг 8

ПРИЛОЖЕНИЕ А

Листинг программы

```
from cryptography.fernet import Fernet
from cryptography.hazmat.primitives.asymmetric import ec
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives.asymmetric import utils
from cryptography.hazmat.backends import default_backend
import base64
import os
from cryptography.fernet import Fernet
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC
import time
import random
import math
from sympy import isprime


def gcd(a, b):
    while b:
        a, b = b, a % b
    return a

def gcd_ex(a, b):
    if b == 0:
        return a, 1, 0
    else:
        gcd, x1, y1 = gcd_ex(b, a % b)
        x = y1
        y = x1 - (a // b) * y1
        return gcd, x, y


def gen_p(q):
    res = 2 * q
    while not isprime(res + 1):
        res *= 2
    return res + 1


def gen_q(L):
    res = "0"
    while not isprime(int(res, 2)):
        res = ""
        for i in range (1, L - 1):
            random.seed()
            res += str(random.randint(0,100)%2)
        res = '1' + res + '1'
    return int(res, 2)



def gen_g(q, p):
    res = 1
    h = 2
```

```
    while res == 1:
        res = pow(h, (p - 1) // q, p)
        h = random. randint(2, p - 2)
    return res



def encr(key, x):
    data = ''
    for i in x:
        if type(i) == bytes:
            data += str(i, 'utf-8')
        else:
            data += str(i)
        data += '\n'
    data = data[:-1]
    cipher_suite = Fernet(key)
    data = bytes(data, 'utf-8')
    enc = cipher_suite.encrypt(data)
    return enc

def decr(key, enc):
    cipher_suite = Fernet(key)
    dec = cipher_suite.decrypt(enc)
    dec = str(dec, 'utf-8')
    res = dec.split('\n')
    return res

def check_sign(keys, r, s):
    x, y, h, q, p, g = keys
    u = gcd_ex(s, q)[1]
    a = (h * u) % q
    b = (r * u) % q
    v = (pow(g, a, p) * pow(y, b, p)) % p % q

    if v == r:
        return True
    else:
        return False


def gen_keys(N):
    C = random.choice(N)
    h = abs(int(hash(str(C))))
    q = gen_q(len(bin(h)) - 2)
    p = gen_p(q)
    g = gen_g(q, p)
    x = random.randint(1, q - 1)
    y = pow(g, x, p)

    return x, y, h, q, p, g

class B:

    def __init__(self, M, i, N, keys):
        x, y, h, q, p, g = keys

        C = random.choice(N)
```

```
        self.C = C

        self.h = h
        self.q = q
        self.g = g
        self.p = p
        self.k_z = x
        self.k_o = y
        self.k_s = Fernet.generate_key()
        self.M = M
        self.name = f"Избиратель {i}"
        self.s = []

    def B1(self, N):
        C = self.C
        #print(C)
        s = 0
        k = 0
        r = 0
        while s == 0 or r == 0:
            k = random.randint(1, self.q - 1)
            r = pow(self.g, k, self.p) % self.q
            s = (gcd_ex(k, self.q)[1] * (self.h + self.k_z*r)) %
self.q
        s1 = (C, r, s)
        s2 = encr(self.k_s, (s1[0], s1[1], s1[2], self.M))
        s3 = (s2, self.M)
        self.s = s3
        print(f"{self.name}:\nВыбор: {C}\nПодпись: {s1}\nПрикладываем
M и шифруем: {s2}")
        return s3



def V1(k):
    m = []
    for i in range (k):
        m.append(Fernet.generate_key())
    return m


def A1(s, m):
    if s[1] in m:
        return True
    else:
        return False



while True:
    k = int(input('Введите количество избирателей: '))
    n = list(range(1, 1 + int(input('Введите количество кандидатов:
')))))
    #k = 10
    #n = [1, 2, 3, 4, 5]
```

```
    m = V1(k)
    print('\n1. V отправляет M всем B до голосования')
    for i, el in enumerate(m):
        print(f"Избиратель {i}: {el}")

    print('\n2. V отправляет A весь набор M, но без информации о том,
кому они принадлежат.')

    print("\n3. B создает свои ключи K_зак , K_отк и выкладывает в
общий доступ K_отк , а также создает секретный ключ (K_сек ), который
нужен, чтобы никто не узнал содержимое бюллетеня до нужного момента.")
    Bs = []
    keys = gen_keys(n)
    print("K_зак и K_отк будут одинаковыми для всех пользователей")
    print(keys[0], keys[1])
    print('\nK_сек:')
    for i in range(len(m)):
        b = B(m[i], i+1, n, keys)
        print(f"Избиратель {i+1}:", b.k_s)
        Bs.append(b)

    print('\n4. B формирует сообщение C, где выражает свой выбор,
подписывает K_зак , прикладывает к нему полученную M и шифрует
K_сек.')
    choices = []
    for b in Bs:
        choices.append(b.B1(n))


    print('\n5. K зашифрованному тексту B прикладывает M и отправляет
A.')
    for b in Bs:
        print(f"Избиратель {i+1}:", b.s)

    print('\n6. A получает зашифрованный текст, по M определяет, что
он пришел от B, но не знает от кого именно и как B проголосовал, после
публикует его')
    valid_ch = []
    for ch in choices:
        if A1(ch, m):
            print(f"Голос принимается: {ch}")
            valid_ch.append(ch)

    print('\n7. Опубликованный зашифрованный текст служит информацией,
чтобы B отправил K_сек.')

    keys_mass = []
    for b in Bs:
        if b.s in valid_ch:
            print(f"Для сообщения {b.s} ключ - {b.k_s}")
            keys_mass.append((b.k_s, b.s))

    print('\n8. A собирает ключи, расшифровывает текст, подсчитывает
голоса и присоединяет к опубликованному зашифрованному тексту C без
M')
    res = []
    for el in keys_mass:
        res.append(decr(el[0], el[1][0]))
```

```
for el in res:
    print(el)
    if check_sign(keys, int(el[1][1]), int(el[1][2])):
        res.remove(el)


res_vote = [0] * len(n)
for el in res:
    res_vote[int(el[0]) - 1] += 1

for i in range(len(valid_ch)):
    print(res[i][0], valid_ch[0])


print('Результаты голосования:')
for i in range(len(res_vote)):
    print(f'Кандидат {i+1}:', res_vote[i])
```