**Министерство науки и высшего образования РФ**

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых»**

**(ВлГУ)**

УДК 004.8

Срок хранения 2 года

ВЛГУ.10.05.04.04.05.00 ПЗ

**ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ**

**«ЛАБОРАТОРНАЯ РАБОТА №4 БЕЗОПАСНОСТЬ ИНФОРМАЦИОННЫХ АНАЛИТИЧЕСКИХ СИСТЕМ NSPK»**

Специальность 10.05.04 – «Информационно-аналитические системы безопасности»

|  |  |  |
| --- | --- | --- |
| Руководитель |  | к.т.н. доцент кафедры ИЗИ Д. В. Мишин |
|  |
| Исполнитель |  | студент группы ИСБ-119 Д. А. Журавлёв |
|  |  |  |  |

Владимир 2023 г.

**Лабораторная работа №4 Безопасность информационных аналитических систем NSPK**

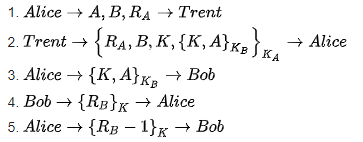
Протокол Нидхема — Шрёдера — общее название для симметричного и асимметричного протоколов аутентификации и обмена ключами. Оба протокола были предложены Майклом Шрёдером и Роджером Нидхемом. Вариант, основанный на симметричном шифровании, использует промежуточную доверенную сторону. Этот протокол стал основой для целого класса подобных протоколов. Например, Kerberos является одним из вариантов симметричного протокола Нидхема — Шрёдера. Вариант, основанный на асимметричном шифровании, предназначен для взаимной аутентификации сторон. В оригинальном виде оба варианта протокола являются уязвимыми.

**Алгоритм:**

При схеме шифрования с симметричным ключом, предполагается, что секретный ключ известен и серверу аутентификации T (Трент) и обоим субъектам обмена: A (Алиса) и B (Боб).

Изначально оба субъекта имеют секретные ключи: K\_{A} и K\_{B}, известные только им и некоторой доверенной стороне — серверу аутентификации. В ходе выполнения протокола Алиса и Боб получают от сервера новый секретный сессионный ключ для шифрования взаимных сообщений в данном сеансе связи, то есть сообщения от Алисы к Бобу расшифровать может только Боб, сообщения от Боба к Алисе расшифровать может только Алиса.

Кроме того субъекты обмена должны быть уверены, что пришедшее сообщение было отправлено именно тем, с кем должен произойти обмен. Боб должен быть уверен, что получил сообщение именно от Алисы и наоборот. Это также обеспечивается протоколом. Предположим, что обмен инициирует Алиса. Будем полагать, что сервер аутентификации у них общий. Рассмотрим реализацию протокола

****

Обмен начинается с того, что Алиса генерирует некоторое случайное число R\_{A} (идентификатор), использующееся один раз. Первое сообщение от Алисы к Тренту содержит в себе имена участников предстоящего обмена и генерированное Алисой случайное число:



При получении этого сообщения Трент извлекает из базы данных секретные ключи Алисы и Боба: K\_{A} и K\_{B}, а также вычисляет новый сессионный ключ K. Далее Трент посылает Алисе следующее сообщение:



Алиса может расшифровать и прочесть сообщение от Трента. Она проверяет наличие своего идентификатора R\_{A} в сообщении, что подтверждает то, что данное сообщение является откликом на её первое сообщение Тренту. Также она проверяет имя субъекта, с которым собирается обмениваться данными. Эта проверка обязательна, так как если бы не было этого имени, Злоумышленник мог бы заменить имя Боба на своё в первом сообщении, и Алиса, ничего не подозревая, в дальнейшем бы взаимодействовала со Злоумышленником. Часть сообщения Алиса прочитать не может, так как эта часть зашифрована ключом Боба. Алиса пересылает Бобу зашифрованный его ключом фрагмент:



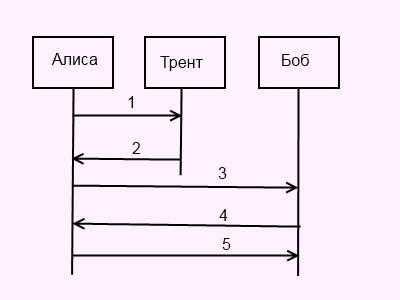
Расшифровать его может только Боб, так как оно зашифровано его секретным ключом. После расшифровки Боб тоже владеет сессионным ключом K. Имя Алисы в сообщении подтверждает факт, что сообщение от неё. Далее при обмене данными будет использоваться сессионный ключ. Чтобы сделать схему симметричной и уменьшить вероятность атаки воспроизведения, Боб генерирует некоторое случайное число R\_{B} (идентификатор Боба) и посылает Алисе следующее сообщение, зашифрованное сессионным ключом:



Алиса расшифрует его и посылает отклик, который ожидает Боб, также зашифрованный сессионным ключом:



**Диаграмма последовательности:**



**Программа:**

**Вывод работы программы:**

1 ALICE -> A, B, Ra -> Trent

1 message = [('A', 'B', 9097)]

2 Trent -> [Ra, B, K, [K, A]Kb]Ka -> Alice

2 message =[(b'gAAAAABkL-rUyXhzSPJAv4KGCMPQKq5Xdbn8bbYfFbSAdx5TQbO1D-R09uxO8MRFsWsnKycMa\_R5Nr9e\_T8p00\_txip0tybWZrkNwYs3-DJUZkupmXxnKs-bkzs2aZTeRTn4-XrCbc-x7daSn5WmXgn0q0UlBy3WDkXTe7rO454UmSChACMxxcgMf5n2HaIaF-xh3nR32PM2I5AwKHJPoevULc7h5LcfGYsSwaT1v3kEjSl1CjWtC9D-XkKKaSuDQJE3\_vvO-cq-7EQSvPHLRaLDr8xcQjhV7RukqekF\_k8KIoBqUohG9yg63DyjQLrsTs5g5JsN8F9Pf1PKzscpIVAiClQVNn4PI4L2\_K8gr77VJgPl0I5zWJ0gD0by1FeqSDjZp3c-WJdsYarWdb2m4SfsZSrBqHEOgtc02CQTTztVgZLuEviuH5TUnqc2s7vj2zFoMIwUg6LKcOwdpx7Ym4dPHJ1HiQrADqGyHscpwIW4rnmuFDFawZ4UwGwSrQYvhh9DuwLKoPlM2Bp1cE\_1cQYevL120FpqiNi54rZD5174IESuyFcjR8UL2vAF7t5x9govPk2274OU5\_7XhqJQwaaTrK21fOwGWyvlr7ooBCfg9maWEA4DCona82HKYYfRHHn3yXdf9ghp3ZghxqTlWqHiiXV-RWl4SGbDExnPiKUt-VynR0ZhCo39BBqHc2sj802piVOLzTu4rHezL6UFrdKdnwwmE5ONfeXqcqK3YDmDad\_9iwsb4UiAvdHFMymFZVEJvhsaXX691bZzIRsUS2fB4igcay1Z17ot1HC5t9O4A6s-QavbeJIRxUsGnlPXrm27gMEJjJUwpTLBcO1xpOHmOoAH\_Tv8FuGp4I9\_pGyNEmZ\_UJAVSsO0K1rsd3wBdGm4AxGv\_zv0txWU1OEC\_p1Ia5uV5Q7NzaeBmtmekLXRga0YYwRN1-5k6e3r6TM891V1FUuAxDkJ5exIhPWJm9P-mxAqlZ378eET2zlZMscyJO8sptIGb7PIDmi7qEBOev7Ej3M-Ouevs1oQpv4iji2P4dzIuyKXpejcUTDZVX\_f8bosnNP6edIQa3yROrN1unQd24P3-2fvKmGPOrCb\_Zue\_8zgKIP1oBvCI-T11Mz8UOyctRYrRi8MCx9WaugWSsyRfku1M2FU5-7kuVAdwXlIJswNTSXPYJKuUJD8uTvW6hWyAIi\_UhcIH-QMT1GnzR7g46iay\_6\_DUeOiwq\_jVy2GmuF8EfH9os4OhvyhsnJhyDpJnNzgjVqbfTWeeS4ywuxVS0KG2k24s1RQemItmg1kJ6PZcXcidhhuAd7DxC3pslX0Vgqc92ibJ9gVP9P-ZeeYiwFTVwAVUjAbcaSXlcgEQ-Mg8Eb3H\_H42oA2cYA3pj\_kG1idZ8PGoF7ErsCX29SB6Q2P0hStYG6jG1fK6ggNyoEWpamyF9Wd-dMD-JacQaKWg8liZ3zBIlWuEqLbkSdKeWu9Ri4y5fXE648L0aYtW1JDADco2VqH19P89jXTYjahc3xO6DKdtAKJvswvz7KkxSRjIe1QKCNe\_nUxnS2aD8MBC2s9BrcQn9BHWElUjJgL7tLxZpHklIWPaOvdKte-Kykv4zbHTRFIo7n7Gj4PzWifKZEQNHNO4TbY916mRbM407-mjHOmbo3UHBoJY0Jrw9Dyrokg1MJrWOEeBf8f1hQc63hLuta2PaXs-d7tUBe4vyX8UavHn9yNOPNalqxy0zSnt9DvfRd-n3l5vLMQlO2wvOOFebfBaVGPDEybq6hOTYgvIVe5eu7M-\_rnyQ0FOEkEHEjxImGuwcDd6KgH5YlUiA7kf3OggvbAnLptST2oV9GzbgYbUDCD06bc7vqFoHyOcMgZx9fxUob-C9dUL-DejD4b9i1A4eaOerUB7sxRbVyqmcxDPHizDB8ViUbS1ztW23ortaqEaofUrw9TnLOcPfgBaVM06B2p7kIFl2sbDL8w2wEGAdUVx8yfIVRDNZzxOuuqklHCp3DFCk5O0JuF88glvyE6q0xKVpnOGzr3wEBy2iDuL3w1OfMKmhV0lNKPMX3X6TKKf28vTZqIDT0aBm76HSVk\_V2hRNMvI0Xo-6V9SMCx4LodudYK3E0f1pd0kpiiK9q6\_VhbvjSfI-YdVBRt\_00p98TebFYHjeCne3cZsQWgR9mJUmiIIaEkVojFBfb1lC2IBnQvpUgDX\_TaLTcGe8vIaMlr0P51MG8tk-Jb2gRqjD1PrNO\_SlULDqB\_Z1BaqGQkiP8D8jrVV\_RDHZnpOYFq70B-wXfbbw6lo27dYs\_WvU-0HgLVY-yZKqk5bXZ2UNnYFQvTGAJwLBhMfgI3paaUWpuC8EmK9zg07lT3LvwKz0k9JBKdCQTItGtVUk1mTUJLSpjYnytvBmi2o4DLydrz8s4U-xSgOZ1-5uRJTn2IQO4J\_jP9gLJDq\_qBz10MjAS9xpYOYlGkuFSSKcXkfAtHH9FH\_jzui-ae5w4BhWs7RkxFVQvsbiiRVfmSeQ-MnJymwJjHSzZ9NHikYDqE0WDa0sEb1OZKt3FTIHs8mE\_mHqjM\_cGQF7iJZJbuAmcjqyhJsbgyxzMDrhZczodvG2nQRMK69Ne1e56axtVIYkJqft8s2-md9PXe\_EhIGSTvm7Fqmw26uD0dUgyDn2BCO4fyb0F0ou4kBliEOq6w-\_YyaU7vM4B4aWMiGxt41ZBcosvmBW39RTw28cpXMBxZr6F47ZUJrmRllaXsYtJH59iJJRtcZja4YDJNvBkMmMG4G3FlJ9Bj-z16l8-k1zKRUdikXJCj8cH95L-jRBzTz9KM2eTQTRsEPxYIJ3fYw-GU5\_Lx6\_6PUvcySqTMeI0Wx4V1OKLjJg5Pf-6-\_nH0RXw\_X\_PAe-uvdXos\_B4F5kqjrCCIZleiDnfttX1qPuwQmJ9Mw3KnW4QF29vp56tYnEGUvAArJeZCT8wlepoh922Cek9lAb31tacG0xwQ137kk6ArPUpzsjVcHEZ10eqLJGzpQT8hPBsK9ovEa-BE9Q-OFx6Mt6TRTGWPUiGX75X03tRbkibHnW0LEV6BxdcnBLQTSveBG9HuJSMcQwllh5LmbAOfKfw7o0mqgwPcZJcZdFCi7J55NFuzI8xB860DZ4TrLPFv4gwiEDxKo9w5tif6U4MzY7mmLpq\_ISVl2Ln0NAxpk1\_nwU0cjX-ZRJjX5ePBVRrnGeXKrGw63c7hLHh-3bHvqlIF2DcQJO1LvSViAUhTkOYXqgCgvK3sQQzWDOWl-N8AqJ6p4lOcZKpJxPtguhTmmQyoctFp7ND8iCDN68a75UTGyY2Py42cB9jMIXzj1Ayws4-y4SUHJ1dPxkKu\_DSH7BCyS2LPE52Lj1kd4zKOxvWHYPRICMGOPrep2KH4-SXfmInfm7KZ5A72LJLR9JUQp7eloIqEj5cFJv6WyUx7ixYOZTMk90VgmZcFxjWiqkrgSa-ffMjLhVaedcehLfN2qKsz9UgssIdHnB5QleUpuTJLX\_mz\_vkJcORMVoQK38Hn7UBAcSDzxSH1KvUx3SlK7SzpHLNyJlK3Vijc-ypYeHsMjRZ4fh\_scuLdHyC3pAsWU8jsZbDdwsA73FDTUCYIO8UbqSde2FepjELWeUZZrcgCCFYREUDO3RBVpboe1ZNXaBDPpazekchtLHufKwxZxdQlgSKuiCUTwo3jMs54aGKIkuDaqNipoYLLBaXmaM068rhadb9QHmmUYr7o\_5kh5zWJ1zGQsd9x6zzYrIHg0w70fClDs2G4jROIijTSKRe-eOBTqUaDQ6k6dE7dHQFFgZjKvG0YKPLtNo35eymNfh9rBIvpTKSfCdXNiGgs3onGMg1RbeeCfRN5hA9xrIMPYPoLKd7sopPKRmAyG4zl3Zwt3ocdD8Zd5bGE2d8Uc3WgbWpqqwrca0byV4SzLYAv4eQB22kM2FQQWRjr74zi2wtEXVj3C09xH-\_4SEsoGzHpXw\_LtSHmA4ncerihMIgKQI8H1uP0f7yn6x3WfXEMA0-qyoPrKHLTVcx3hJ-eNjyeLeGmhubN8eOJY\_kDk9rCOZiA9KI7U0HKeRGaD1CWaxHd1I4WMyUs2q7p\_lA4nsCmS7bPliYIEeOa4W1t7PUxAk9Uk9jRna3QwyNMLx5DssPVRSVy28k4-7vuVGrX65lRn38VYOHAvK2xb3i92I35Jict4aPHyZF1yelgyz1vf7c9xJHr6xI82k3uD7tk8\_2PwII731D2QQjOvOSJqSCWWiMhgIaVYv8fVzIEVPKWZaCpHPhYKiVg4So-xkdLz535KGxFFEpK8DrQ1C-WIi6\_bld0MqxoZLPQ9j\_2O\_FgxKBuOfaAFPBZColBzJTQ7FB5OFv0mFsMy\_k5zsTzT0-qM3eF3KqymJkRzHOpRVm-D83bxMPD7btBwRw3D9Cgdj1rRqukyuKCMSiPI3JDPt4N9ovEZXwWXj8sbZ3Cph0a7Ftdi1JyTlwtCHUVbX\_h4OBViv9byWcaBqrcJiDjXCOn\_f0CNbUNLiByWXpeHKQU\_soSXRFiw2QXxpweLKeUM5KNXuy6yX4X0JU61i1rOVxfKUYYK1ku2vG5VP79T9lI1A6O1b5mrAHdq-MV241xMtddNUjlUxaQQqX04w38fSqLqvBJQbpIR0LAmFD\_tfwK8\_\_WBAkft7idChR6OjIiEBdZnzGx-swxL5awlbdjpAGZnDGkDFhWyszzMdDy-uzSYueBHM4gmYAIpmuRetJB3iJJJFOYEnKGyR-WLXpPRxlt8Fn5PzfrCTkSNJn\_aYVi0W6vz2OuY0nm83YhScpdmcC2YKZ1QzmFyjxDviZfLIJ6H-MpeLaOeJYrVFcLpysae9pBPIqaT-vFfQ0HsIG7HqQW-lwns-dHwtJ2Up9QSg30QKYNTSHBhNs\_BoUGtxdB1ghc-54YYYye-PrzCb\_-wbNRI6m-\_mj2DHJx9Lr22oQz3cX9ThJihpSu6q2gA9bmvc8td1PlY1e74MjDXH1NYAs84y2iEyU6t9ebMW4w5a3AlJHNqzvj1jGhHZFW7TqeMrmrpMAQRr6ISgdkPC5aZYduMD07-A91o23IBsZio8I81Z1endCBzo7PpYQScT8yPDShrehWldlpwRpkPzZBacZh5L\_lVRC0Nvo\_qRVWOE646OS5Ho73-X77FCgvO1Bh7H-faJ69lbO3z8qJqhkIncghDoENkg1XW6seK\_EK0JD-pH1W7-SrIQ8M8nf4NJwWKCzC97qSGT7Va3vcN111x-NolNHI3nWjDYDEvcJuXEVjpYD03rWhx2tHgfvysSGmcRnNLuoyOSqPm\_P6xgnpUGB-EvPJQeXq0QNM4OH1EW\_gBc0ehb5YKeQz98DRC4rszuxFo40Lj78Nr5VMNolT20blK-DmD9rv6qTYhwTuWghv6LuU79t-oeCVP05UufSOpjvfgJi3nLDm60tR2wOx8L-UyhbRRUC\_sz4f1HVb36Z98n5kMkjpbrVt4CBYw0ZdDdZC-6HB9GwU2SfUzpMvkIp2tQfF\_IjjavaFezTO8wDA5TM8TrtFrHMoj3OIgbfym39CbYzHueCdMPZbtRyFLxzwtVUdpJSmRwcfg19x0VzMGq1d2G7q6SHzjtkOtAUkh8AR3dBwE90PwtdCZczdd2QDocth8PcbiQkizZOqbH94V9tL-V0O\_wj\_Mwdg-2NvdaDZPtSBj3BTuWrMM2yu3NA02ZO-jmCpTbPEt5aI46detOahZg0DuqPJpAfK94b6VaaEJE9vMLOQLP616ucRfRPuZvAwc4z0uS\_Y-efjP1VD9448-2KFnWpShHXmq\_Ux8wGZ2c0XQ1pjdVER7yn0LGGJ4ZRDLw-RicRUnAKWSWQ135Ct4pPdyqSRA7U4Fu1XW98UXJeHVCzS0a0CBg5GwsGCw3k4J3TNWhQ0LNvNdE18yVuSSoSeHIgMsNCoYnjgh6UpPCNjgMhND9zE06yMJg-1bYgTCJPiKP2ny-\_QoZIzfnUlxR0Zu\_wLAmPmv7cTfCGyjQv\_FAVCya2PRNLdgAyrBkBWtTeV3QagNyvWVhXwKrlqiCVehqrfLPXagYQiZfVh3Z8pzjCe\_rjKo0ZYyhPYM6JXA450U7zMXis1Id64HmOnF7Oxi8zPNfLDNFm\_ROeDNAVCEoVHD7Ia0Rv35i1lUsZSFPlew2X3bmvqOGZ\_XPhPEmNFDmtcG\_w-BqsLtqKhr36V3OhoDlmZPnnBoS10hzQrlqJLV28-6x2\_89NoYSvGDRvhxilgP20yaWYsYigyeVsEfzU6CAKlkFWCNTFRYyk0xk-aXzlutJ-cpPAvrE2qgi6Mg\_I-t9gbKRU3iatLiEGN72mEF8P01KFufmhLGQ1OFd-UpnXMvFLG87kSBDUSs3Cbm-9Np35Fwj3YhgEa1f9RQM4CXeTGVqelt-jSpWkyvuC2Z6bV7aANWZ3D2trhtvdhCDeKt2XTb5ifVYP2Z9T5ezOzaqttqII428-ihc-UbxxfBpMSn2FCKxKA42PyBUnoZFr79YDq8NthKLom6ag5HlGzlMnjuN-tn3Q83\_GZVu6mMMA5aX74RUPdNomKbbIsloNBUJID\_e9QxMwhSvwy3h3tGqN0arFYREuTazmP3gCjjS787oB4Xhd-Is1mQSm7Pqxl-zkuJj5GPtFyZmmoJWGYvoX0oGTqmKToaBkuCn9QalDMdCM8wSurfWm2IXih3zVwWakNjl3jJQSRR0P0yyljJ\_tIf-NBnctLDe2HyZVQWjOtCjTsK5guqCuKcJIHX5y2\_\_HiWRXJE91kA8t3BaN95Irdy9dsKyZ4zLgrV-uUHj2YeQ11CyrEtjNkatpiNRHziRvb4Roer779AljyG4usvAdPE7gsHsuu5d5muuvCS0gVOwO1K5RFLzE6iaHaLppqrJegXbkuqJsmq5bOViT\_v50njUNH98CaRmrHy2a3PZlAFEL6ksDbZJUFVv8YFaBq616CP4GA9xvzdNtWatIU5R7\_26sEXHa6blljhSEDaqonCifi-1qbBwnfyMZ6yzdO9llbqbGpO8cH004S7lCIIKyHbPEDpzvkNXnpUCfNPtOfioO6vqLjhA\_ziUHgIXKbaSDbAy2FPxeykwhpbV9m3ekJU4yIlpVkT582aN5vJu9buYsZT35n4QegTjHhIE9U-PDN0ILCyVJlvpVcycOJZ66wsU3f-3wW4Dfh1Ah9bQ0AU45C28CQcoFTiPPn0vo6N6nzLtzYhndHoyQAXxp-QVE6BKErtsG2c0TlDWFU6XSkKR94tnZ8wj3-2p54OhP82t\_ubU-JhyR\_C06Nj9AiT2omnNdXf13OKSnb-8to6fzwPJ06jMQozfFzdrhf7HIrt6R1AWJBaPbYrd115IavBbNFdz30WaT2VzwWr7s\_v2ZB0KWprvchSEFOaynArDh4dn6Yc1yAPSuEQ6e06S76MHzUiOOPK27ZcYxCKWcHIvNSurPmbYwdmGv6MpLVJool92S7fOsToLgi6LTtjISA\_GwNfOh0SJcv\_LYRapq33DEQwNHMqpM-D1J0oNAJHM6cWT7-9IrzdThnFhqtR2N9ti5qlPs1DgJwTOsYEYiPGnociuMYH7PGA\_un-wpx6TQ4772wl-HGDEkV3eKQV7s962ByS-VXydjFNiWLIG1WP93wtpKrdBQ6nQ9PT7IJuwRjPjQMqsLrGLUtb8L-e-RsJOi7xYBbz7QmICPxBdvscqZP4cKnf3qT2pZOACHdGmRlMQY7uPQgS42rnCqbh\_F-r1mMjMLvTJGIrPzLnUCIiYOZt176ZgLcIRavf7V9NFr5OKhPNjEtgpMn-fEBdPTDWF3Qk\_avpuXD4QnhIjt-VHB7gtCZVenyy8tlXjnCV7tFuub5kXlMeY50-H0pEbRbA7GXVK9rW\_wX2vWb9IYmIzoDvpbHisFdI9qYCAs5g1Uu2YYCd\_pQHF4wVTnsBUKEP6Tum4IpKghcqttNx-hkD0y1VsxA4t9x2VaRvRleaC38GOmEKnhLz5ANoXBk3TiiXG0vFlNcmBTLjh0M1Q2hH6a6j5hmNNSfsU147TWvgosI2mu8fzaAG2fB1kNpR\_3EeOzZTpQo1RI0Fb9Cus8onLEMnL0sVlolxFCKdpM4D4KyTP2C0QoV7kCEnFZ2ibSEDWMNOJogz4qdjJEGwBvCDMPb4CJp2VQNQkIs7Evn1vlmPFvA8jHKWbxPMLFnZy96ih09RAYRCebVS2WR9DOiU1omxtA4vqrdWU22rV2bGEPj938E7T9hya\_gQpqt-mW1yuHxxw3XL7IjpnY4eqhbcAxeeDdy6XAEVods1UQIGJeK0cuAHawx4vUZkk8LSmtch4inC2K3pGNmCFvkkZh9K4L\_Iol9RTOjY3TOqVo5nS6MwG7Oy1ugGH\_Ky14HANvy6GoEzJ04cjGtGRn4sMx16XZ\_1pwsloYlg\_HjsLDfQmmvML-u0qlsJTQCZsRVhY5SVqMdmqf0MmovOPCL0Ep2BnmR1tGG8VKg04R1OGWXa7syL9CJPL25W6JyHCD7zykggE8FUYmtbyZABLP9kiSKtwmTIrmSBDRlPFQ8fvLX6xMIpv8DWgLJRDLmsYbpRYhgR632mmNTfbxKrSEB8PqvlK9n8FijTVLJV5Sf\_TYApuvYutcetizza8jHuGsEIoN3VW1LtbH\_GfDgI7HqR5sA9jOtu1dvRhvVA1amciKI5D4LcXQIJ9KQJZJjBO5y9B34E4briibxOG\_bVdgilar1TLNKzh7dTvXeAnyPtXCjzc480JXEOvrbtcgICabVfGM-PbN6LeTUu6GI5IfKrSrMYCEK7yAXV600UqyMp9B40txJhKoruvg3SgOy1x593ZEeDQPW7i-gNll\_Mtk3f-Ly8jbC-rjqMRx6\_Qmg-uKpFKoCIkYVaDPWdCWkvRcZsR\_\_UiClS2MqwbUcboQUDJhTeVJHymaUVQJZ3goPn1mWV3eQjL\_Pz8KHy9Ljz9xQ2paU8s5g-B549Vf0OuDq6ssfXvFT8ehaOW57\_8GPGGj1\_rRE6JLOB3-LkBba0cFRKSZ7jYjPPipKL\_SZ-K5254aupt4APM5iSa7rvdVEAdW24514HCBmuirjdjA0yBygx4bTslT95tzSIGwz3sT6geHT7VU2opq49zfDHLv\_Ej9IW8VlWSNgZBDFIWxchMwIIP5o9qVoTd1yNVk23ya7gh8tMv04JdHnnjMechGDZz2zWGQJHPtYrPGHuTG2YNo-\_b8fhnwe8RLmeZ8EKGFKFd1nHcPM5q3DoSonOcBEU6iDaBZxebVO6X8ve9Um8S0ZiqBI0dkTrGuoFgh3BVB7BpZGhoWspTGiAYEshMDOqs363nbfhPeYe5bnFDu4EWhceBKvBRxBgGA1EaBH-6LWDT7fYCRkZFK7SDhfSIGBjBaP0ARlfoCkD5JwasPER4LBTvN37xcR9SHb7-pNKKt6sVKMIBT77PRF36xJNw9of\_wOLUgjLvzFpVnQ30cHzbBMk-we\_n\_1qTn50M0Tc08J5Mo5YFuUAGs5OW4tcLA6dL2hZgt0D\_0NoLuMmTH-e5yilcX-2Ii4oFoKItfbYdfvPcVZfSspJkKfcH\_89XOQZqaO16mHP0XwswbEHDAS7yjAW5hcLIeYakmFHSfUCUDkfq-V5LsRnbOqRXmCHzh98Kyq7nU7Pvd4C3Y24-CagiHjm8Fq1iJYbAy2Zy19B35gIBlbEl7BBsSwzJfwJr1A6POdT4LjsPNvIYKA9wNP1l4bFp2T5gIdXCgGeLxfpKDwl4shQeLfc1-bEdkG5N8USEexVYD9tnEC9Nl8K\_iRj4V1SAChLKHSe5HwYZmUTdTuhQmAOd5zGv9P2ue\_KeTyV69w5q9mMYORUfUy-NbjqOJSxbrD6zYZEooAeLMLKvdVuBt5H0-xD6MUlTDxuoD3RX360U-zNvYKqYS6GmxBtck3\_1aTk7I8mKAfIg2hqyoRORxT9BrGwsg299uBXYzPiF\_CwfSOB2nkS-Nz6ujpLJiGdmD2BAi6r4rUTJUHrtXyFDaqzfOKTLt1HSfweUAUqhXFn9ILkloEgEPBtII6xjJxY0e192R4sj-bLeeqOqpKUfCOm5tNllQePkaKs\_nieLm8mK2b28uvMsxJNuTJe67OdjyYYR7Uzdi3lB\_euBkBwiPIbaQMROOa\_1EA8BHuwkP906nostNcGl7hVd4zwcar2hICwFza8euWSEwTJ26lMn3FAsJMq5aaErFyVkcYf5GD6SZntve3r\_Ha2XhQh86m-NIoR0\_K1wGFKN8PKTnySJBzmvs3gEJym3QDsfqdO-v9QNhOSn8VEzXFK9C\_toGN3Wzfx85A4-asP\_bUk-sJ8abfsbb4ZaZmdJ-fS-wARH-ZiwKtgsNaQvvIuSP6c\_5jsg8C68qBHohZVV-hMCw4Ain\_AlMcROI4z-cPA6S3W7RnlzXNdbXYXD0MYTkG3hiPtN-18QezzS36uRs1wrSjPymI6I\_B0JbcXsuyedZovbNfTjYtXQIyXFmN7UgI\_lORFvNTMTaqJDwoqp3kxp\_DqQX1x3nq\_kI0ZJ0vbftsO\_Pb\_SAwmkqabK1Xg8Z10lYhJouV3T5FPNDoeOQa84EmmDOjl\_BllP3ISEGbtxgJFEcY7AYCKVgqI-YeRPkKj-Fmsj\_sxFJx\_7OWK8eRKdu68lk6BLnhQ7HnX7Irz-VAi2ipKw-W8NsYGLIzA9PQpKXonH67lc9Eyx5pXjDFqxv-jB9A3x-BWlFXiTlKFonPe2TqW\_OYzcsQdd9FcO0wxCMAmBkpoUR0GkZSeu0Yq5XFy-MXLSqHhbQeSdGXGmWNFMSqwxVoyhMTKDp10\_VnXBgFOCoG0LGbS20Z-7pf0vkBqnI\_eI6Y-0\_t-\_VyHDG6CcAl6mHwap7X41Ips7XUmT\_JwimlbnFHwB8ctb2w1KaKnRB-5nTkRLSsa08Vk9ccvKESThblO8F0g00UFCNCJYLPqWyOirWrMFlRRORUfWs\_-jTFlm\_nuPpb-0Fr\_jgX4wtiplUPVGEPwAQDVdQ7jkEVX4FJ37kwm1zH9SbgY87kFx9gFfodHK\_5KUMoGS5gj3IvMSK94gYOKNr3Ozl3zM6EGSyA530gkRa52DL2OVNPoeT48qzZsxRxd-WX7oKYVwjrP8TfSK2qgx1jnWaqZaCvYjB0\_uKMJAF\_qmUvzCAAOwIiqjkAhCFtZ0FFfdZ2FNMLatnKhFX-zvI-ZYKUycKZVLmCFcUDYvOalEQnUQAw-vd2A7ljiZboqojj\_irEGHK1XYJ2gbLuRXYZfQpoLgt\_MfidQjuA9K2rCZiHQ-ShZx6svc3SPNcfB1qvjP7n1r8DtDYt-nSbx6VHF4m5emXVlUCzYGGZK8DSEccATaO\_XZcpxJVPB1shy3pNpaamYPqxj6daieF3tDpdLsLwYelFSza-bPdMXXkplxPpA1PXH6pXQQX9nCb5da\_DjQL3\_SKlPKAlzXc8bQURrshRP4QPnFgYYzNemmOu\_Q-lNX6z34mHMaZxKoZF3H3UvEIq2NPIqw2UARM1rcbvgz53ydDw8SO52zgEjPLlpn8QjoWjGgZly97eZR4HkRGqhyOzHXLaMbEkhscxOtZm5Dzy94m99OIEKcNSkF-JSLja3heaAs2XhTp7nbhT-oB9HBqYAFaydnjTX-0f67wApfZQvqt5jhPp9Yg4pGYXqMlbRyWRega0GVo5CT\_z2RCtTVzjWOwD6nzKE3Ux1ejgkuKx18x8-Ftir3N0IBtTGw4mR5NBk03wCN1E0okVjeCMhjPM9NihNVX9GwX7X4hUKnSY4ti1tByQDiISQ\_vlvP1TAt72THCzQOY5k77BZlWZ\_y28q-EvCkGjnJSMJXMjU13Tr-ET\_XSENCrJHInRNLMW14roeq46ydgk\_KwKkLkdjeW66qmIsx6-h9bg46gdxLMY30W36n8k1sZ6isZ4ZnuAIdezlMe1I73YmMzlqqUDYPTW5JpBkMWJfHZIRHBwjrtYCOkj-oO5CuPcOLnEW\_A5Pxw6ep9Qzpgscgc9DmHzvBLLZQNuqAllhhCg-MV2PG1vXYhVHMd9nLuwTIvLhDt8mkxxrfnXi9H0EdurV9tRRT3stpWdO3G5HelXgNx2b4EOlVIUVOMX6qkX70ASLYERUF2PlHN89Zd8FvaglwXSZ2ovBLGkzN\_54NpuW\_aglq8WnBjDLBxr6-uAp5b1Z6lw9P7W\_3CNkt4PtCkAOVZxUgYTtZUDTo5pA1pBOdKw941LZIXsIYhASVsXnk=', b'gAAAAABkL-rUAngQ-LWHAC1rjQbPijwGWuPlqf0x2acRD38gLrD5l0mbx9ji3\_RwaADMioICKeHAxmf-MG12MTBkFTqKLJDhQQ==', b'gAAAAABkL-rUiqgrXKoENFXnPY5p1PBDEii9NJmN8fj7x7hrfFB2KguTvMD6YbUdU0S-9dPxGuLGJFqiE4HtkTROaLFWB6q3LoWJ44Kzx\_CL\_wPEwv\_Kq292sL2uT9P9VOeWFDEx7yYI', b'gAAAAABkL-rUVNz-DX5qwC2Vb85iZLpxlct1uXtncV1q1LsBeWou4Bp2-baqydA8pdrBd6p2ee3IKrdj6QtXTFwvEGBZnFPh3DzL8zlVKYz7k7731ezoOhsB-SHtatkwJPdNEh4fRyINSzVTEqx74SQy4VlR0EYLpWc26BHiMdKOYkBiKmefYQV5WdjoA6f33819H4gvoujStOZKb\_D8qT-bruzB\_MZjVvMdoxU0LY6q58hqq7SjICKoh3xyQZTEV0t4uJgd-En4', b'gAAAAABkL-rU4TfkmjLQoyHVSArwviFvKjFB4Li23cSl8mJp-LlQ\_9\_6yFbToNpwFMphUNsgkDXZkwzw96PhVvPmZ4Xalj-Qf2BDO25L302O8NRaLDNfQi5yN4exy4t9YkJHSYjdkescieClmyUDtaMpfbNCS-kA79dp\_W3i1cQorL5xllLF4CBea-MMEX1BehyVMtVJTE7hHp3m3CSrARNHHmtodRcRbg==')]

3 Alice -> [K, A]Kb -> Bob

3 message [(b'gAAAAABkL-rUZckZMKd7ro4D\_5eeJLrOVtygKLJCSakjNu0pHJOsQDrEJ4zIwyEKV-9xrfQM5zzL8LUomTGuLy7j2RJ\_KQLgwfE3qneRay\_usV6MmJmAFrxjtljJTqpzll0yZZN2VSxe', b'gAAAAABkL-rUBK-HM\_kMoWA8xy8hCbuT8H9LH5qWSDUkgKdCetYL7aYR3nDEXF7U8o66GJPRmSMdJpbrLK-RgkgHhDlTp0\_vAA==')]

4 Bob -> [Rb]K -> Alice

4 message = [b'gAAAAABkL-rUVeVlSsj7BALctRBE0B4KpBvz9qoM6-\_stgvby8GGCfmsVdLDvP\_O-1xvEE9-SYia0SlTzbyh39yOIpd-fs18pA==']

5 Alice -> [Rb - 1]K -> Bob

5 message = b'gAAAAABkL-rU2hvU20-QNsGKpgLwliDTuCpio389BffN5SL-zEwDCaqIurVON4ofhZeHxyL\_\_ZT0zHtgohJK4KmL5JbiuVWuAA=='

**Приложение:**

import random

from cryptography.fernet import Fernet

TrentBD = {'A':b'XH2buFNIPF9VEZ17eqqTX95a\_fPn\_4BFYZTIqJYGpYI=', 'B':b'zzo52-ZqRWH8gyJe\_1KSiQKwOwxwp1d57L0N9zlt7cw='}

AliceBD = {'key':b'XH2buFNIPF9VEZ17eqqTX95a\_fPn\_4BFYZTIqJYGpYI='}

BobBD = {'key':b'zzo52-ZqRWH8gyJe\_1KSiQKwOwxwp1d57L0N9zlt7cw='}

def f1():

nameA = 'A'

nameB = 'B'

key\_Ra = int(random.uniform(10, 10000))

AliceBD['Ra'] = key\_Ra

AliceBD['nameB'] = nameB

print(f'1 ALICE -> A, B, Ra -> Trent')

print(f'1 message = [{nameA, nameB, key\_Ra}]')

return nameA, nameB, key\_Ra

def f2(nameA, nameB, key\_Ra):

key\_A = TrentBD[nameA]

key\_B = TrentBD[nameB]

key\_K = Fernet.generate\_key()

fernet\_A = Fernet(key\_A)

fernet\_B = Fernet(key\_B)

print(f'2 Trent -> [Ra, B, K, [K, A]Kb]Ka -> Alice')

sA\_key\_Ra = fernet\_A.encrypt(bytes(key\_Ra))

sA\_B = fernet\_A.encrypt(bytes(nameB, 'utf-8'))

sA\_K = fernet\_A.encrypt(bytes(key\_K))

sB\_K = fernet\_B.encrypt(bytes(key\_K))

sB\_A = fernet\_B.encrypt(bytes(nameA, 'utf-8'))

sA\_sB\_K = fernet\_A.encrypt(bytes(sB\_K))

sA\_sB\_A = fernet\_A.encrypt(bytes(sB\_A))

print(f'2 message =[{sA\_key\_Ra, sA\_B, sA\_K, sA\_sB\_K, sA\_sB\_A}]')

return sA\_key\_Ra, sA\_B, sA\_K, sA\_sB\_K, sA\_sB\_A

def f3(key\_Ra, nameB, key\_K, secret\_B\_K, secret\_B\_A):

fernet\_A = Fernet(AliceBD['key'])

if bytes(AliceBD['Ra']) == fernet\_A.decrypt(key\_Ra):

if AliceBD['nameB'] == fernet\_A.decrypt(nameB).decode('utf-8'):

AliceBD['key\_K'] = fernet\_A.decrypt(key\_K)

secret\_B\_K = fernet\_A.decrypt(secret\_B\_K)

secret\_B\_A = fernet\_A.decrypt(secret\_B\_A)

print(f'3 Alice -> [K, A]Kb -> Bob')

print(f'3 message [{secret\_B\_K, secret\_B\_A}]')

return secret\_B\_K, secret\_B\_A

def f4(secret\_B\_K, secret\_B\_A):

fernet\_B = Fernet(BobBD['key'])

key\_K = fernet\_B.decrypt(secret\_B\_K)

nameA = fernet\_B.decrypt(secret\_B\_A).decode('utf-8')

key\_Rb = str(int(random.uniform(10, 10000)))

fernet\_K = Fernet(key\_K)

secret\_Rb\_K = fernet\_K.encrypt(bytes(key\_Rb, 'utf-8'))

print(f'4 Bob -> [Rb]K -> Alice')

print(f'4 message = [{secret\_Rb\_K}]')

return secret\_Rb\_K

def f5(secret\_Rb\_K):

fernet\_K = Fernet(AliceBD['key\_K'])

Rb = fernet\_K.decrypt(secret\_Rb\_K).decode('utf-8')

Rb\_1 = Rb.join('1')

Rb\_1 = fernet\_K.encrypt(bytes(Rb\_1, 'utf-8'))

print(f'5 Alice -> [Rb - 1]K -> Bob')

print(f'5 message = {Rb\_1}')

nameA, nameB, key\_Ra = f1()

key\_Ra, nameB, key\_K, secret\_B\_K, secret\_B\_A = f2(nameA, nameB, key\_Ra)

secret\_B\_K, secret\_B\_A = f3(key\_Ra, nameB, key\_K, secret\_B\_K, secret\_B\_A)

secret\_Rb\_K = f4(secret\_B\_K, secret\_B\_A)

Rb\_1 = f5(secret\_Rb\_K)