**Swiss post protocol**

**Overview:**

This overview is drawn from Haines et al [5] paper. Global parameters are created by the trusted setup (SDM) at the beginning of the system. Through the SetupTally protocol, the electoral board and mixing control components (CCMs) collaboratively generate a public key while exchanging a secret key. Voters get voting cards with voter credentials and verification codes generated by the trustworthy print office and CCRs using the SetupVoting protocol, which also contains the cryptographic data (CMtable) required for return code recovery with CCRs. Voters receive voting cards during the voting phase, and they cast votes using a web-based client that is relayed to CCRs over an untrusted server. Voters receive jointly computed return codes that are derived from valid votes. After voters verify it with a ballot casting key, CCRs verify its authenticity by working together to compute and return a vote cast return code. Unconfirmed votes are then eliminated during the Tally Phase, and CCMs hosted by Swiss Post mix and partially decrypt the encrypted votes in order. Auditors use VerifyVotingPhase and VerifyOnlineTallyPhase to confirm the evidence provided by control components. The last mix and decryption is finished by the canton CCM, and auditors use VerifyOfflineTallyPhase to confirm the proofs. In this paper we use the terms CCM and mixer interchangeably.

**Attacks:**

**Lack of signature validation - Attack on individual verifiability:**

Haines, T., Pereira et al. [5] found a vulnerability in the authentication phase of the protocol specifically in `validateSignature` and `validateChoiceCodesEncryptionKey` methods. These tests confirm that the input is signed, but they don't confirm the signer's identity. The adversary can infer the identity of authorised parties since they have valid signing keys. Additionally, the system does not verify that the keys are utilised for the intended purpose or that the subject field of the linked X.509 certificate matches the expected party.

This vulnerability is further discussed by swisspost in their Gitlab issues board [6], The election’s public key is supplied exclusively to the voting server during the configuration phase, signed by the administration board secret key, and is not directly received by the control components from the setup component. For zero-knowledge proof verification, which is essential for individual verifiability, the Return Codes control components (CCR) during the voting phase require the election public key ELpk. It is possible for a malicious voter to offer authentic Choice Return Codes for an invalid vote by taking advantage of the lack of sufficient checks.

**Attack on individual privacy:**

The decryption process occurs in a single round, starting with mixer1 mixing and partially decrypting the list of ballots. Subsequently, mixer2, mixer3, and mixer4 continue this process sequentially. This involves auditors in two stages: first, they act as an intermediate spot check before the offline CCM4's involvement. If mixer1, mixer2, and mixer3 provide consistent data, auditors prompt the Election Board to reveal the final part of the decryption key to mixer4. Finally, auditors verify the correct decryption performed by mixer4.

Cortier et al. [7] found a vulnerability that can exploit two functionalities of the protocol, no verification at the CCM’s level and producing relatively small-scale results than the handled ballot sizes. The potential threat actor for the described attack is a malicious insider or a malicious government agent among the Control Components (CCMs) or the Voting Server. Specifically, the attack involves collusion among three dishonest CCMs and a dishonest Voting Server.

The attack goes as follows [7], after obtaining the *k* ballot boxes for decryption, an evil Mixing component1 creates an additional one, Bk+1, which just holds an individual’s ballot. After that, mixing component1 submits the *k+1* ballot boxes to Mixing component2 in the usual manner. Genuine, mixing component2 combines and decrypts the *k + 1* boxes, then sends the results to mixing component3, who repeats the process. The malicious Mixing component3 eliminates elements corresponding to Bk+1 before providing the data to the auditors. Auditors find this to be accurate, which leads to the disclosure of the Electoral Board's secret Mixing componet4 key. In the end, Mixing component4 (malicious) works with Mixing component1 and Mixing component3 to finish decoding Alice's ballot. Crucially, the *k* "valid" ballot boxes can be decrypted by Mixing component4 covertly. Because the dishonest Voting Server may be aware of the relation between ballots and voters (e.g., IP address), this attack enables three conspiring dishonest CCMs and a dishonest Voting Server to discover an individual's vote and, consequently, the vote of any voter of their choice. By adding malicious ballot boxes, [7] the attack can be expanded to discover the votes of several voters while staying undiscovered unless the additional boxes cause the process to delay. Since all communications go through the dishonest Voting Server, Mixing component3 (working with CCM2) may also be truthful.

**French legislative e-voting protocol:**

The French legislative e-voting protocol specification [1] is not completely open but briefly provides data about practical deployment and overall functionality. Debant and Hirschi [2] Reverse engineered the specification working and their description mainly focused on the newly found channel attacks in FLEP.

**Attack**

**Lack of correspondence of voters to their ballots**

Debant et al. [2] found a vulnerability with the FLEP system which lacks the correspondence of voter’s receipts to their ballots. The FLEP system ensures the election integrity by using the receipts. To ensure their ballots have been placed in the voting box accurately, voters can choose to send a server a copy of their receipts. Then a third party verifies the decryption zero-knowledge proofs (ZKPs) to ensure the accuracy of the election results. These procedures act as protections against manipulating any one vote or the election's total result. The authors [2] observed a nuance in the JavaScript code of the FLEP voting client. In particular, the voting client's computation of the ballot references may not match the ones that are provided to the voter. Since the references of the voting servers are not converging with the voting clients in all the instances a malicious actor can exploit the channel attacks.

An attacker who gains access to the communication channel can alter the election results by modifying and discarding the votes. This disrupts the individual verifiability characteristic of the protocol.

**Russian federal remote e-voting protocol**

In the 2021 Russian parliamentary elections, two new e-voting protocols were deployed. Here, we cover one of the two protocols called the Russian federal e-voting system. However, the documentation for the protocol is lacking and only provides a high-level overview.

Participants involved in this protocol are listed below [3] voter refers to a Russian Federation citizen who meets the criteria for voting and is listed among e-voters. A voting device can be any device with a browser and internet connectivity, which is used by the voter to cast their vote. Organiser manages and oversees the voting process. Registrar performs the authentication of voters using a voter list and responsible to provide blind signatures for the voter’s public keys. Vote collector interacts with blockchain for the issuance of ballots and publishing the encrypted votes to the block chain. While Tallier consists of the Decryptor and voting storage.

**Protocol:**

The protocol description here is based on the protocol summary provided by Vakarjuk, Snetkov et al. in [3]. For creating key pairs for the initial signature scheme, the Registrar and Organiser work jointly. After that, the Tallier receives the public keys. The Tallier then authenticates the signatures on all association from the Registrar and Organiser, which are signed using their private keys. The Registrar creates a commitment key, forwards the public key to the Organiser, and generates an RSA blind signature key pair. The Organiser creates an ElGamal key pair with the secret key split through Shamir Secret Sharing in front of Election Observers and the media. The original key is erased, and the key shares are moved to an external storage device.

The public key is sent to the Organiser by the Decryptor, which also creates an ElGamal key pair. In order to create smart contracts, the organiser uploads public keys and election information to the blockchain. The registrar then gets the VoterList from the blockchain. Voter codes from the VoterList are used by the Registrar to compute commitments, which are then uploaded to the Blockchain. A final encryption key is created by the Organizer and uploaded to the Blockchain. To decrypt, both the keys, which are derived from Tallier's and the Organizer's public keys, are needed. The Organizer and the Tallier are unable to learn midway results until the Organizer's split key is put together until tallying.

The Registrar uses the federal identity code to confirm eligibility after the voter authenticates on the online voting site. The voter gets an authorization code, which they he/she will input into the portal. The Registrar provides an RSA blind signature, which the Voting Device uses to create a GOST signature key pair. The VoterList is updated, relevant information is noted, and it is published on the Blockchain by the Registrar.

On a digital ballot, each choice is designated as a bitstring, and the voter selects them during the voting phase. Every choice is encrypted independently using ElGamal encryption. To ensure that each ciphertext is either 0 or 1, and that the total of all values does not exceed a certain limit, the Voting Device generates proofs for each data. The voter signs the transaction, which is prepared and transmits to the vote collector along with ciphertexts, proofs, and key pairs. Then the Vote Collector uploads the transaction to the Blockchain, adds key pairs to its database, and verifies the signature, ballot integrity, and transaction uniqueness. The Blockchain smart contract publishes the transaction after additional ballot and signature verification.

**Attacks:**

Vakarjuk, Snetkov et al. performed the security analysis based on nine defining criteria [3], The results from his study infer that the voting system cannot assure a voter to cast-as-intended verifiability as voters cannot confirm if their digital ballot choices were casted correctly. Although voters can verify their voting transaction in the Blockchain retrieval, they cannot ensure that their vote remains unaltered, and this lacks the recorded-as-cast verifiability. The decryption of individual votes from the Blockchain is impossible due to the publication of only a portion of the secret key after the tallying phase. [3] This allows a malicious voter who controls the voting device to alter the casted vote and the encryption scheme encrypts the falsified vote and this attack goes unnoticed since the voter can only verify his vote casting but not the actual contents of it after it has been casted.

Additionally, the following attack depicts how a malicious government can tie a voter’s identity to a casted voted, [3] Collaboration between the Vote Collector and the Registrar during the authorization phase could result in the Voter's IP address and browser metadata being stored by a corrupt Registrar. In the same way, this data may also be recorded by the Vote Collector during the voting phase. As a result, the Vote Collector receives information of the Voter's encrypted vote together with IP address and metadata, and the Registrar obtains knowledge of the Voter's identity, IP address, browser, and device details. The attackers can tie an encrypted vote to the voter's identity by cross-referencing this data. But to decode each vote, it is necessary to compromise the Organiser and Decryptor (HSM module) in addition to the Registrar and Vote Collector (which a malicious government can easily do during the setup phase).

**Moscow Internet voting system**

Gaudry et al. [8] demonstrated how easy it was to get the private keys using the public keys because of the small key sizes and when this issue was fixed in the updated version in which the developers sticked to the single level ElGamal. However, the authors of [8] found another vulnerability in the newer version, which lets the adversary to count the number of votes casted for a candidate.

**Attacks**

The description below is derived by Gaudry et al [8], the first ElGamal encryption process uses a cyclic group G of order q produced by g is used. Choosing a random decryption key (sk) from Zq yields the matching public encryption key (pk), which is determined by pk = gsk in an ElGamal keypair. Encg,pk(m) = (a, b) is a random process that represents the ElGamal encryption of the message using public key pk and generator g. Encg,pk(m) = (a, b) = (gr, pkr · m) is the outcome of equivalently selecting an integer, r, from Zq. The related decryption function Decg,sk(a, b) is provided by b · a-sk = m and uses the secret key sk corresponding to pk.

The triple encryption in Moscow’s system involves applying ElGamal encryption successively with three different parameter sets, initially on the message m and then on the first part of the successive ElGamal ciphertexts. The three group generators are represented by the symbols g1, g2, and g3. Ballots are encrypted and decrypted using three sets of keys: sk1, pk1, sk2, pk2, and sk3, pk3. The message m is encrypted using a three series of ElGamal encryptions, and the quadruple of G1 × G2 × G3 cipher texts is formed.

The authors of [8] used three different software products which can implement the discrete log computation, they include: SageMath, Magma, and CADO-NFS. Both SageMath and Magma used GP/Pari, was initially tested on a personal computer with a 4-core Intel i5-4590 processor running at 3.3 GHz and 16 GB of RAM. The computation, which utilized a linear sieve index calculus method, but a rerun with Magma on a 64-core server node was successful, completing the calculation just under 24 hours. CADO-NFS, an implementation of the Number Field Sieve retrieved the first private key in 425 seconds, 507 seconds for the second key, and 314 seconds for the third key.

The second vulnerability concentrates more on deputy IDs.The deputy IDs have a 50% chance of belonging to the quadratic residue group Qp, since they seem to have been chosen at random without any discernible arithmetic pattern. This is true for any element of F\*p. Given that deputy IDs are 32-bit integers, this probability is still unbiased. From this kind of attacks, unless the voter casted his/her ballot for a less popular candidate, one can infer her choice.

**References:**

[1] Voxaly Docaposte. Partial specification of the flep, 2022. Available at the link <https://w8t9w2j6.stackpathcdn.com/wp-content/uploads/VOXALY_LEG2022_Verifiabilite_Specifications.pdf> obtained from <https://www.voxaly.com/vote-parinternet-pour-les-francais-de-letrangerdans-le-cadre-des-elections-legislatives-2022>

[2] Debant, A., & Hirschi, L. (2023). Reversing, Breaking, and Fixing the French Legislative Election {E-Voting} Protocol. In *32nd USENIX Security Symposium (USENIX Security 23)* (pp. 6737-6752).

[3] Vakarjuk, J., Snetkov, N., & Willemson, J. (2022, June). Russian federal remote E-voting scheme of 2021–protocol description and analysis. In *Proceedings of the 2022 European Interdisciplinary Cybersecurity Conference* (pp. 29-35).

[4] Post, S. (2021). Infrastructure whitepaper of the Swiss Post e-voting system. *E-voting documentation*.

[5] Haines, T., Pereira, O., & Teague, V. (2022, September). Running the Race: A Swiss Voting Story. In *International Joint Conference on Electronic Voting* (pp. 53-69). Cham: Springer International Publishing.

[6] <https://gitlab.com/swisspost-evoting/e-voting/e-voting/-/issues/1>

[7] Cortier, V., Debant, A., & Gaudry, P. (2021). *A privacy attack on the Swiss Post e-voting system* (Doctoral dissertation, Université de Lorraine, CNRS, Inria, LORIA).

[8] Gaudry, P., & Golovnev, A. (2020). Breaking the encryption scheme of the Moscow internet voting system. In *Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers 24* (pp. 32-49). Springer International Publishing.