Breaking and Fixing Vote Privacy of the Estonian E-Voting Protocol IVXV

Johannes Müller[0000−0003−2134−3099]

johannes.mueller@uni.lu
SnT, University of Luxembourg
Luxembourg

Abstract. We revisit the e-voting protocol IVXV that is used for legally-binding political elections in Estonia from a privacy perspective. We demonstrate that IVXV is vulnerable to attacks against vote privacy in those threat scenarios that were considered for IVXV originally. We explain how to improve IVXV so that it protects against the privacy issues we discovered.

1 Introduction

More than 15 years ago, Estonia became the first country in the world in which voters could regularly cast their ballots for political elections over the Internet. The initial e-voting system used for political elections in Estonia was a rather naive system whose security relied on trusted voting authorities; in particular, that system did not allow for auditing the correctness of the election result. Not least due to the attacks against the initial e-voting system discovered by Springall et al. [11], Heiberg et al. [4] proposed a new e-voting protocol designed to mitigate trust in the voting authorities and to allow for external auditing. This protocol, which is called IVXV, has been used for legally-binding political elections in Estonia since 2015 until today.

Our contributions. In this work, we revisit IVXV from a privacy perspective. We discovered that IVXV (described in Sec. 2) does not guarantee vote privacy of all voters, which violates a fundamental right of Estonian voters (Article 1(2) of Riigikogu Election Act). More precisely, in Sec. 3, we present efficient attacks against vote privacy of IVXV that exploit the malleability of the underlying encryption scheme. In Sec. 4.1, we show that the assumptions of our attacks are realistic or within the original threat scenario of IVXV, respectively. In Sec. 4.2, we elaborate on real-world implications of the privacy vulnerabilities. Eventually, in Sec. 4.3, we explain how to fix the privacy issues of IVXV presented in this work.

Responsible disclosure. We shared our insights with representatives of the Estonian election authorities on August 23, 2021, and discussed our findings with them as well as members from the system provider (Smartmatic-Cybernetica)/a main author of IVXV in an online meeting on September 2, 2021. Following this meeting and a further online meeting, Smartmatic-Cybernetica acknowledged the existence of the privacy issue of the IVXV *protocol* [4] that we present in this paper and declared their intention to fix this issue as proposed in Sec. 4.3. However, they claimed that in the actual *system* used for real-world elections, "there are quite a few physical and organisational safequards implemented" that would already mitigate the risk of these vote privacy issues; we discuss their argument in Sec. 4.1.

2 Protocol Description

In this section, we recall how the IVXV e-voting protocol works on the conceptual level. We restrict our presentation to those phases of IVXV which are relevant for the privacy attacks described in Sec. 3, emphasizing that these attacks also work against the full IVXV protocol, as presented in [4].[1]

2.1 Protocol participants

The *Election Organizer (EO)* is the administrator of the election who determines the list of candidates/choices, list of voters, etc. EO is also responsible for generating the public/private key material used to encrypt and eventually decrypt the voters' choices.

The *Vote Collector (VC)* is active during the ballot submission phase in which it collects the ballots submitted by the voters. VC checks incoming ballots for eligibility of the respective voter and stores all ballots according to the time at which they have been submitted.

The *I-Ballot Box Processor (IBBP)* is started once the submission phase has closed. IBBP takes as input the ballots collected by VC, verifies its correctness, and removes all data which is no longer needed for the subsequent tallying phase (e.g., voters' signatures).

The role of the *Mixing Service (MS)* is to anonymize votes before they are decrypted by EO. MS takes as input a list of ciphertexts from IBBP, re-encrypts and shuffles this list, and returns the resulting mixed ciphertext list to EO.

Any external party who wants to verify the integrity of the election result (i.e., whether the election result corresponds to the votes submitted by the voters) can run the *Data Auditor* program which takes as input all data published by the election authorities and verifies its correctness.

2.2 Cryptographic primitives

IVXV employs the following cryptographic primitives:

[1] To be more precise, in our presentation, we simplified the checks carried out to verify eligibility of voters, and we completely omitted the voters' individual verification procedures.

A *digital signature scheme* $\mathcal{S} = (\mathsf{KeyGen}_{\mathcal{S}}, \mathsf{Sign}, \mathsf{Verify})$ for signing ballots and for signing all public data output by the election authorities.

A *homomorphic public-key encryption scheme* $\mathcal{E} = (\mathsf{KeyGen}_{\mathcal{E}}, \mathsf{Enc}, \mathsf{Dec})$ for encrypting the voters' plain choices. In the IVXV implementation, $\mathcal{E}$ is instantiated with the ElGamal PKE scheme [2].

A *proof of shuffle* π_{Shuffle} which is generated by MS to prove that it shuffled its input ciphertexts correctly. In the IVXV implementation, π_{Shuffle} is instantiated with Verificatum [14].

A *proof of correct decryption* π_{Dec} which is generated by EO to prove that it decrypted the final ciphertexts correctly. In the IVXV implementation, π_{Dec} is instantiated with a Schnorr-based NIZKP [10] to prove knowledge of discrete logarithms.

In the original IVXV paper [4], it had not been specified which security properties these primitives need to provide precisely but the respective instantiations suggest that the signature scheme $\mathcal{S}$, the encryption scheme $\mathcal{E}$, the proof of shuffle π_{Shuffle}, and the proof of correct decryption π_{Dec} are supposed to be EUF-CMA-secure, IND-CPA-secure, and non-interactive zero-knowledge proofs (NIZKPs), respectively.

2.3 Protocol phases

The IVXV protocol is split into the following phases. In the setup phase, EO creates the key material. In the submission phase, voters can submit their ballots. In the tabulation phase, the ballots submitted by the voters are anonymized and decrypted. In the auditing phase, which can be executed at any point after the election result has been announced, the correctness of the data output by the election authorities can be verified externally.

Setup phase. The protocol assumes that there exists a public-key infrastructure (PKI) of public verification keys for individual voters. The Election Organizer EO determines the list of eligible voters which are identified by their respective public verification keys. EO also determines the set of valid choices C (i.e., party lists with individual candidates). EO runs the key generation algorithm of the public-key encryption scheme $\mathcal{E}$ to obtain an encryption/decryption key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}_{\mathsf{Enc}}$. The list of eligible voters, the set of valid choices C, as well as the public key pk are made available to everybody.

Submission phase. Each voter V_i who wants to vote for some choice $\mathsf{ch} \in \mathsf{C}$ encrypts her choice as $\mathsf{c} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{ch})$, then signs the ciphertext c with her secret signing key ssk as $\sigma \leftarrow \mathsf{Sign}(\mathsf{ssk}, \mathsf{c})$ and submits the resulting pair $\mathsf{b} \leftarrow (\mathsf{c}, \sigma)$ to the Vote Collector VC.

For each incoming ballot $\mathsf{b} = (\mathsf{c}, \sigma)$, VC verifies whether σ is a valid signature for c w.r.t. a verification key vk of one of the eligible voters. If this is the case, then VC stores b together with the time at which it had been submitted. Voters can re-vote multiple times.

Tabulation phase. After the submission phase has closed, VC forwards the list of ballots to the I-Ballot Box Processor IBBP who first verifies that all ballots in VC's list were signed by eligible voters only. Afterwards, IBBP removes the ballots of all voters who have also submitted a paper vote. Eventually, IBBP extracts the last submitted ballot of each voter (who did not submit a paper vote), removes the respective signatures, and stores the resulting ciphertexts in a list B_1.

The I-Ballot Box Processor sends the list of ciphertexts B_1 to the mixing service MS which then re-encrypts all ciphertexts in B_1, shuffles the resulting re-encrypted ciphertexts uniformly at random, and computes a proof of correct shuffling π_{Shuffle}.

After that, MS sends the resulting ciphertext vector B_2 to EO which first uses its secret decryption key sk to decrypt all ciphertexts in B_2 to obtain the final result res, and then computes a proof of correct decryption π_{Dec} to prove that it decrypted B_2 correctly.

The tuple $(B_1, B_2, \pi_{\mathsf{Shuffle}}, \pi_{\mathsf{Dec}}, \mathsf{res})$ is the output of the tabulation phase, where the list of plaintexts res determines the raw final election result. Eventually, EO publishes a "sanitized" version of the raw result res from which all invalid choices, if any, were removed.

Auditing phase. The Data Auditor DA takes as input $(B_1, B_2, \pi_{\mathsf{Shuffle}}, \pi_{\mathsf{Dec}}, \mathsf{res})$ and verifies whether π_{Shuffle} is a valid proof of shuffle w.r.t. the lists of B_1 and B_2, and whether π_{Dec} is a valid proof of correct decryption w.r.t. B_2 and res.

3 Privacy Attacks

We describe two attacks against vote privacy of IVXV [4] which we will call *shifting attacks* and *encoding attacks*, respectively. Both attacks exploit the homomorphic property of the encryption scheme, that is employed in IVXV to encrypt the voters' plain choices (recall Sec. 2), to create maliciously generated ballots that depend on honest voters' ballots [3]. Both attacks slightly differ in terms of the underlying assumptions and their impact: in comparison to the shifting attack, the encoding attack requires qualitatively stronger assumptions but it allows to break privacy of several voters by submitting a single malicious vote. In this section, we focus on the purely technical description of the attacks and refer to Sec. 4 for a discussion of their assumptions and implications.

3.1 Background: Homomorphic Encryption

Recall that a public-key encryption scheme $\mathcal{E} = (\mathsf{KeyGen}_{\mathcal{E}}, \mathsf{Enc}, \mathsf{Dec})$ is *homomorphic* if both the message space $(\mathcal{M}, \cdot)$ and the ciphertext space $(\mathcal{C}, \odot)$ are (algebraic) groups and the encryption algorithm is a homomorphism between these two groups, i.e., $\mathsf{Enc}(\mathsf{pk}, m) \odot \mathsf{Enc}(\mathsf{pk}, m') \in \mathsf{Enc}(\mathsf{pk}, m \cdot m')$ for all $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KeyGen}_{\mathcal{E}}$ and all $m, m' \in \mathcal{M}$. For example, the ElGamal PKE scheme [2], the one employed in the IVXV system, is homomorphic.

The privacy attacks presented in the remainder of this section are based on the following facts.

Note 1. Let $\mathcal{E}$ be a homomorphic public-key encryption scheme, and let $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KeyGen}_{\mathcal{E}}$. Then the following statements hold true:

- For all $m_1, m_3 \in \mathcal{M}$, we have that $\mathsf{Enc}(\mathsf{pk}, m_1) \odot \mathsf{Enc}(\mathsf{pk}, m_2) \in \mathsf{Enc}(\mathsf{pk}, m_3)$ holds true for $m_2 = m_3 \cdot m_1^{-1}$.
- For all $m_1, \ldots, m_n \in \mathcal{M}$ and all $\alpha_1, \ldots, \alpha_n \in \mathbb{N}$, we have $\prod_{i=1}^{n} \mathsf{Enc}(\mathsf{pk}, m_i)^{\alpha_i} \in \mathsf{Enc}(\mathsf{pk}, \prod_{i=1}^{n} m_i^{\alpha_i})$.

3.2 Shifting Attacks

The idea of the *shifting attack* is to submit a ballot which contains a vote for ch' if and only if the targeted voter V submitted a vote for ch. If ch' is an unpopular choice, then the adversary learns whether or not V voted for ch by checking whether there exists a vote for ch' in the final election result.

Assumptions. We make the following assumptions:

1. The attacker can learn the ballot of the voter V whose privacy he wants to break.
2. There exists a (valid) choice $\mathsf{ch}' \in \mathsf{C}$ which is chosen by none of the (honest) voters with high probability (see 2nd paragraph in Sec. 4.1).
3. The attacker can control one voter.

Impact. The attacker can check whether V voted for ch.

Program. The attacker runs the following program:

1. Submission phase: Learn ballot $\mathsf{b} = (\mathsf{c}, \sigma)$ of voter V.
2. Submission phase: Submit ballot $\mathsf{b}' = (\mathsf{c}', \sigma')$, where $\mathsf{c}' \leftarrow \mathsf{c} \odot \mathsf{Enc}(\mathsf{pk}, \mathsf{ch}^{-1} \cdot \mathsf{ch}')$.
3. After election: Check whether $\mathsf{ch}' \in \mathsf{res}'$.

3.3 Encoding Attacks

The idea of the *encoding attack* is to submit a ballot which encrypts a unique encoding of the choices $\mathsf{ch}_1, \ldots, \mathsf{ch}_k$ submitted by the targeted voters $\mathsf{V}_1, \ldots, \mathsf{V}_k$. The attacker then checks for all possible combinations $\mathsf{ch}'_1, \ldots, \mathsf{ch}'_k$ whether there exists an encoding of these choices in the final result. If the attacker finds such a combination, then he knows that $\mathsf{V}_1, \ldots, \mathsf{V}_k$ voted for $\mathsf{ch}'_1, \ldots, \mathsf{ch}'_k$, respectively.

Originally, the concept of what we call encoding attacks in this paper goes back to Pfitzmann's seminal works [8, 9]; here, we use a generalized version of Pfitzmann's attack presented in [6].

Assumptions. We make the following assumptions:

1. The attacker can learn the ballots of the voters $\mathsf{V}_1, \ldots, \mathsf{V}_k$ whose privacy he wants to break.
2. The attacker can learn the raw election result res.
3. The attacker can control one voter.

Impact. The attacker learns how $\mathsf{V}_1, \ldots, \mathsf{V}_k$ voted.

Program. The attacker runs the following program:

1. Submission phase: Learn ballots $\mathsf{b}_1 = (\mathsf{c}_1, \sigma_1), \ldots, \mathsf{b}_k = (\mathsf{c}_k, \sigma_k)$ of voters $\mathsf{V}_1, \ldots, \mathsf{V}_k$.
2. Submission phase: Submit ballot $\mathsf{b}' = (\mathsf{c}', \sigma')$, where $\mathsf{c}' \leftarrow \mathsf{c}_1^{\alpha_1} \odot \ldots \odot \mathsf{c}_k^{\alpha_k}$ and $\alpha_1, \ldots, \alpha_k$ are integers chosen uniformly at random.
3. After election: Let $\mathsf{ch} \in \mathsf{res}$ be the invalid choice in the raw election result res. Return $\mathsf{ch}^1, \ldots, \mathsf{ch}^k$ such that $\mathsf{ch} = \mathsf{ch}_1^{\alpha_1} \cdot \ldots \cdot \mathsf{ch}_k^{\alpha_k}$ holds true.

Efficiency. The computational complexity of the encoding attack is $\mathcal{O}(n^k)$, where n is the number of possible choices and k is the number of targeted voters. Hence, in practice, several dozen voters can be targeted efficiently. If the attacker only wants to check whether $\mathsf{V}_1, \ldots, \mathsf{V}_k$ voted for $\mathsf{ch}_1, \ldots, \mathsf{ch}_k$, then complexity reduces to $\mathcal{O}(k)$.

4 Discussion

4.1 Assumptions

We show that the assumptions that are sufficient to execute the privacy attacks against IVXV (Sec. 3) are realistic or within the original threat scenario considered for IVXV [4], respectively.

On learning targeted voters' ballots. The IVXV protocol was explicitly designed in such a way that it "allows to outsource the vote collection task to a third party, as the correct operation of this party is verifiable by voters, third-party auditors and auditors nominated by the election organizer itself" [4]. We argue that the fact that they claimed that *any third party* could perform this task implies that VC should also not be trusted in terms of vote privacy. Since VC receives all incoming messages, an attacker who controls VC can learn all submitted ballots (even if he cannot manipulate them undectably). Hence, the assumption on learning targeted voters' ballots is within the general threat scenario considered for IVXV originally.

In our online meeting, Smartmatic-Cybernetica stated that, unlike in [4], in the system used for real-world elections, the task of VC was not executed by a third party but by a service which they claimed to be protected by physical and organisational safeguards. Even if this is the case, we think that it is undesirable having to trust a party (VC) for vote privacy whose original purpose/role is not to protect vote privacy (but to simply collect votes). Furthermore, since collected votes are not published in IVXV, it is questionable whether all parties (voters, observers, etc.) can have the same view on the data being processed, which is a crucial property for secure e-voting in general (see, e.g., [5]).

On the existence of unpopular choices. For the shifting attack, we assume that there exists at least one unpopular choice (and that the attacker knows a priori that this choice is unpopular). In what follows, we show that this assumption is realistic. The electorate of parliamentary (Riigikogu) elections in Estonia is divided into numerous districts. For each of these districts, there exist party lists which contain several candidates of the respective party. Altogether, each voter can vote for one candidate of one party list in one district, which results into a large number of possible choices in total and a granular public election result. It is therefore not surprising that in the last Riigikogu election [12] in most districts, there existed several candidates that received only one vote in total (most likely, the vote of the respective candidate), or even no votes at all. It is also realistic to assume that an attacker knows a priori for at least one of these candidates that she/he will not receive any votes with high probability (except for her/his own vote).

On learning the raw election result. For the encoding attack, we assume that the attacker can learn the "raw" election result which also contains invalid plain choices. Unlike in many modern e-voting systems, the Election Organizer EO only publishes a "sanitized" version of the raw result from which all invalid choices have been removed (see [13]), which may contradict our assumption on first sight. However, there exist several parties which learn the raw election result anyway. For example, according to [4], "trusted representatives of political parties, foreign research groups or even local civil activists" may fill the auditor's role. In order to be able to audit a run of the IVXV system end-to-end, an auditor needs to obtain the raw election result; otherwise, it wouldn't be possible to verify the proof of correct decryption employed in IVXV. It can therefore not be ruled out that one of those parties who audit an election has malicious intents. Indeed, the IVXV protocol was explicitly designed in such a way that a possibly corrupted auditor is not able to learn how individual voters voted: "[d]ue to the re-encryption mixnet used, the malicious auditor could not break the ballot secrecy" [4]. This proves that the assumption on learning the raw election result is clearly within the threat scenario considered for IVXV originally.

On corrupting a voter. It is realistic to assume that the attacker can control a voter because the attacker can be an eligible voter herself. For the same reason, in case some voters collude, the attacker can execute the privacy attacks multiple times to target even more voters and break their vote privacy.

4.2 Implications

In democratic elections, vote privacy is a *universal* right: the individual vote of *each single* voter must remain secret. All voters, not just the majority of them, must have the right to express their true will without facing personal negative consequences. Indeed, it is particularly important to protect the privacy of those voters who favour less popular parties/candidates because these parties/candidates as well as their supporters are more likely to be threatened than those (who voted for the) ones in power. Additionally, in each election, there exist voters whose individual choices are particularly interesting to single out for non-political reasons. For example, a party of an ongoing trial may want to learn the judge's personal political orientation in order to increase its advantage in the court or to blame the judge being biased.

Furthermore, even if vote privacy of only a fraction of the electorate can be broken, no voter can be sure whether or not she is among the targeted voters. This observation can be exploited to coerce many voters at the same time (not) to vote for a specific party/candidate because each coerced voter will likely follow the coercer's instruction if she knows that the coercer can randomly check whether or not she obeyed.

The mere possibility of these attacks can cause a significant bias in the election result and thus undermine the legitimacy of the government elected.

4.3 Protection

Fortunately, it is straightforward to protect against the privacy attacks presented in Sec. 3, as described next. In the submission phase, each voter computes a NIZKP *of knowledge* π_{Enc} which proves that the voter knows plaintext ch (and randomness r) such that $\mathsf{c} = \mathsf{Enc}(\mathsf{pk}, \mathsf{ch}; r)$. This mechanism is commonly employed in modern secure e-voting systems (e.g., Helios [1]) because, in this way, the ciphertext c is no longer malleable if the correctness of π_{Enc} is verified before further processing (see, e.g., [3]). If IVXV is modified accordingly, the homomorphic privacy attacks presented in this paper are no longer possible.

We note, however, that this modification may not necessarily protect against *all* possible attacks against vote privacy of IVXV. In order to ensure that the IVXV protocol in fact provides vote privacy, a formal reduction proof is necessary. However, the current presentation of IVXV in [4] does not allow for such a proof because the security properties of the underlying cryptographic primitives are not specified precisely (recall Sec. 2) and the overall protocol model is partially underspecified. Furthermore, the threat scenario for vote privacy (as well as the ones for verifiability and coercion-resistance) needs to be stated more explicitly than in the original paper [4] and in the current online documentation.

Recommendations. We recommend to improve IVXV as follows:

1. Voters in IVXV use a NIZKP π_{Enc} as described above.
2. The (fixed) IVXV protocol is presented in full technical details.

3. The threat scenarios for all security properties (verifiability, privacy, coercion-resistance) are described explicitly. Security is formally proven.

We hope that the insights on vote privacy from this paper as well as the ones on verifiability from [7] will make electronic elections in Estonia more secure.

Acknowledgements

I thank Jan Willemson for his open-minded discussions about the vote privacy (issues) of IVXV as well as his critical remarks on a preliminary version of this report. I also thank the anonymous referees for their helpful feedback.

Johannes Müller was supported by the Luxembourg National Research Fund (FNR), under the CORE Junior project FP2 (C20/IS/14698166/FP2/Mueller).

References

1. Ben Adida. Helios: Web-based Open-Audit Voting. In *Proceedings of the 17th USENIX Security Symposium, 2008*, pages 335–348. USENIX Association, 2008.
2. Taher El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *Advances in Cryptology, Proceedings of CRYPTO '84*, pages 10–18, 1984.
3. Rosario Gennaro. Achieving Independence Efficiently and Securely. In *Proceedings of the Fourteenth Annual ACM Symposium on Principles of Distributed Computing, Ottawa, Ontario, Canada, August 20-23, 1995*, pages 130–136. ACM, 1995.
4. Sven Heiberg, Tarvi Martens, Priit Vinkel, and Jan Willemson. Improving the Verifiability of the Estonian Internet Voting Scheme. In *E-Vote-ID 2016, Proceedings*, volume 10141 of *LNCS*, pages 92–107. Springer, 2016.
5. Lucca Hirschi, Lara Schmid, and David A. Basin. Fixing the Achilles Heel of E-Voting: The Bulletin Board. In *34th IEEE CSF 2021*, pages 1–17. IEEE, 2021.
6. Shahram Khazaei and Douglas Wikström. Randomized Partial Checking Revisited. In *CT-RSA 2013. Proceedings*, volume 7779 of *LNCS*, pages 115–128. Springer, 2013.
7. Olivier Pereira. Individual Verifiability and Revoting in the Estonian Internet Voting System. *IACR Cryptol. ePrint Arch.*, page 1098, 2021.
8. Birgit Pfitzmann. Breaking Efficient Anonymous Channel. In *EUROCRYPT '94, Proceedings*, volume 950 of *LNCS*, pages 332–340. Springer, 1994.
9. Birgit Pfitzmann and Andreas Pfitzmann. How to Break the Direct RSA-Implementation of Mixes. In *EUROCRYPT '89, Proceedings*, volume 434 of *LNCS*, pages 373–381. Springer, 1989.
10. Claus-Peter Schnorr. Efficient Identification and Signatures for Smart Cards. In *CRYPTO '89, Proceedings*, volume 435 of *LNCS*, pages 239–252. Springer, 1989.
11. Drew Springall, Travis Finkenauer, Zakir Durumeric, Jason Kitcat, Harri Hursti, Margaret MacAlpine, and J. Alex Halderman. Security Analysis of the Estonian Internet Voting System. In *ACM CCS, 2014*, pages 703–715. ACM, 2014.
12. Valimised. https://rk2019.valimised.ee/en/voting-result/voting-result-main.html (accessed 23.08.2021).
13. Valimised. https://www.valimised.ee/en/internet-voting/documents-about-internet-voting (accessed 23.08.2021).
14. Verificatum Mix Net (VMN). https://www.verificatum.org/html/product_vmn.html.