Analysis of the Finding

Document identification

Customer	NSW Electoral Commission
Project Name	iVote® 2019
Subject	Review of the attack described in the report "*Faking an iVote decryption proof*" by Vanessa Teague, Associate Professor, dated October 2, 2019 ("the Report")
Document date	21st November 2019, Updated 28th November 2019
Author(s)	Scytl R&S department: JP, Review: SC

1 Executive Summary

A recent new report from Vanessa Teague proposes that the decryption proof implemented in the iVote®[1] voting system used in New South Wales for the March 2019 election "*is susceptible to the same attack (the author) identified already in the equivalent part of the SwissPost/Scytl system*". This paper reviews the Report, the requirements to reproduce a similar scenario as the one present in Switzerland, and the practical limitations preventing the execution of a similar theoretical attack as in Switzerland. We note that there are no nonsense votes detected, nonsense votes being an indicator of a successful attack.

This analysis goes further than the Report, as the Report discusses the ability to use a cheating decryption service, whilst the Scytl response covers how such an attack would be detected based on other system features.

2 Introduction

In March 2019, a report describing a finding related to the decryption proof used in a Scytl's new Swiss Post voting system was published by Sarah Jamie Lewis, Olivier Pereira and Vanessa Teague[2]. This finding was based on the use of a non-adaptive zero knowledge proof (Fiat-Shamir transform) for the decryption process implemented when the same servers implement both the Mixing and decryption processes. As described in section 2 of the same report, the use of this transform "*(…) only works in a non-adaptive setting, in which the proof statement is given in advance to the prover and verifier.*", otherwise it is necessary to include the full statement (i.e.: the encrypted vote) in the proof (i.e.: to use

[1] iVote® is a registered trademark owned by the NSW Electoral Commission (NSWEC), and for the remainder of this document will be simply denoted iVote.
[2] Lewis, Sarah Jamie; Pereira, Olivier and Teague, Vanessa. "*How not to prove your election outcome: The use of non-adaptive zero knowledge proofs in the Scytl-SwissPost Internet voting system, and its implications for decryption proof soundness*", 2019. (the "Swiss Report")

Strong Fiat-Shamir). In the case of Switzerland, mixing and decryption were implemented in the same server and therefore the adversary can interact with the mixing process to hide any manipulation of the attacked vote, thus making the scenario adaptive.

The NSW iVote voting system was live when the finding above was reported, and it was also implemented using the same decryption proof software by Scytl. However in the NSW iVote voting system the decryption process was implemented in a completely isolated machine (disconnected from any network), which is different to the implementation in Switzerland. The input received by the decryption machine is the output from a mixing process which was also implemented in a different isolated machine, and the data transfer was done physically using a removable media (e.g.: a USB drive). Therefore, the encrypted votes used as input for the decryption process (the statement) were known in advance by the prover and verifier (via removable media). This makes the scenario non-adaptive: any attempt to fake a proof using a manipulated ciphertext is detected when using the output of the Mixnet to check the decryption proof. For this reason, it was considered that the finding in the Swiss Post voting system was not relevant to the iVote voting system.

The Report analyses the decryption proof implemented in the iVote voting system used in New South Wales for the March 2019 election and the author demonstrates that the decryption proof of a vote could theoretically be manipulated if similar conditions as in Switzerland can be reproduced in iVote. In practice that means that the attacker must subvert the Mixing and the Decryption isolated computers and make them operate in a similar way to those in Switzerland – in other words operate as if they were deployed in the same machine. As in the Swiss scenario, the Report mentions that any manipulation ends up generating a vote with “nonsense that would not be counted”. Due to this creation of nonsense vote content, any practical implementation cannot hide evidence of the attack, and no nonsense votes were detected at the end of the NSW March election.

The following attack analysis examines the possibility of a practical attack based on the assumption that the attacker can reproduce a similar environment to that in Switzerland, using the iVote infrastructure. The analysis considers the complexity of the attack, its implementation assumptions, its scalability, and the fact that it is always detected.

The conclusion is that the attack is extremely difficult to implement in practice, requires collusion between multiple actors, breaks a number of other controls and would be easily discoverable due to the presence of the ‘nonsense votes’.

Full protocol documentation, including updates, are available via the Scytl Online Voting Source Code Review Program:

- https://www.scytl.com/en/AccessiVote2019

3 Theoretical attack analysis

3.1 Requirements for a practical attack

The attack requires two main changes in the iVote voting system:

- Manipulation of iVote components and their software
- Changing the operational workflow of the election finalisation process

Both are described in the following sections.

3.2 iVote voting system components

In the NSWEC system the decryption process is performed in an isolated server which does not execute a mixing process. The mixing process is similarly performed in an isolated server which does not perform a decryption process.

Due to this implementation practice, it is not possible to hide manipulation of the input data to the decryption server, as the manipulation will be detectable when validating that the data leaving the Mixnet matches that input into the decryption server.

Using as reference the theoretical attack scenario described in the report on the Swiss Post voting system, the attacker needs to compromise several different voting system components (voter devices and the first control component). In the NSWEC implementation of iVote the number of voting system components increases and includes completely isolated systems. More concretely, in the iVote scenario an attacker needs to control the following components:

- As many voter devices as voters whose vote is targeted by the attacker
  - The attacker cannot successfully manipulate any vote in the decryption process unless he/she learns the randomness used in the encryption process by the voter device[3].
- The isolated decryption server
  - The attacker needs to control this server in order to remove and replace the original iVote decryption software with the attacker's software that performs the attack and generates fake proofs. We explain the reason why below.
- The isolated mixing server
  - The attacker needs to control this server and remove and replace the original iVote mixing software with the attacker's software that uses information generated by the attacker's rogue decryption software to manipulate the encrypted votes and allow the validation of the fake proofs.

[3] The randomness of the vote is necessary to allow the Mixing server to make a mixing proof that hides the manipulation that is required by the decryption process (as explained by the authors in section 3.3 of the Swiss Report). Without this randomness it is not possible to recreate the Swiss adaptative scenario. When the Swiss adaptive scenario is not fully created the attack will be always detected when using the mixing output and the decryption fake proofs, or when verifying the mixing proof using the input of the decryption process. The decryption server must know from which votes the randomness has been leaked, only to identify the votes that will be the target of the attack.

In the Report it is stated that one of the requirements is that the mixing and decryption servers are running Scytl's software. However, analysis reveals that this assertion is misleading. The attack described cannot be executed by the original Scytl software, as it implements the correct cryptographic operations. The attack requires a replacement of Scytl code in both servers to implement the mathematical operations mentioned in the Report to manipulate votes and generate the fake proofs. The attack is *not based on attacking a bug in the code* rather *it is based on exploiting the vulnerability of the non-adaptive decryption proof when it is used in an adaptive scenario*.

The same concept applies for the leakage of the randomness from the client, as the original Scytl software does not implement a mechanism that facilitates the guessing of the randomness used by each voter. Due to this the attack requires the manipulation of the Scytl software used in the voting client, or alternatively some form of malware that could capture or control its randomness and share it with the attacker.

In order to emulate an adaptive scenario, the workflow must change as detailed in the next section.

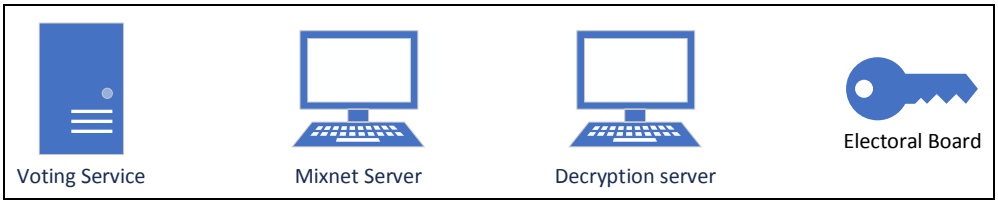
3.3 iVote system workflow

As stated in the previous section, the attack not only requires the removal and replacement of the server software to perform the mathematical operations, but also changes to the operational workflow to facilitate the non-adaptive scenario required for the attack. This workflow change is relevant for the attack implementation – in order to understand clearly it is important to see the normal workflow, and a workflow to support the adaptive scenario.

3.3.1 iVote standard workflow

The following describes the operational workflow in a standard iVote environment. It is important to be aware of this process for a normal election to detect abnormal behaviours of the actors present in the process (as in any traditional election, via paper or electronic processes).

Note there is an air gap between each of the Voting servers, the Mixnet server and the decryption server. That means that information between these servers needs to be transported using removable media, such as DVDs or USB devices.

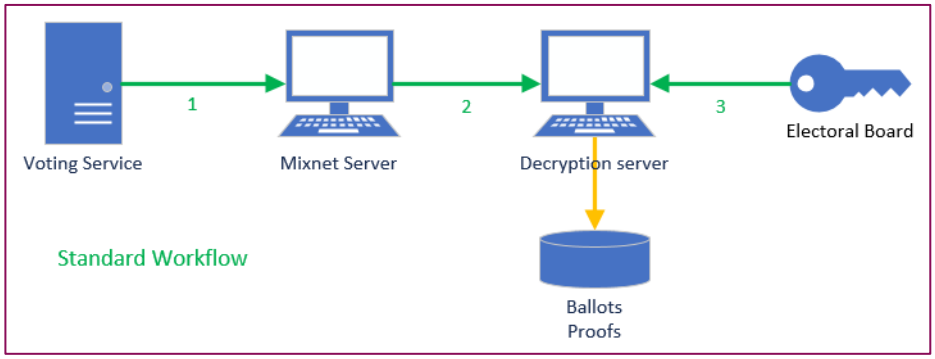


In a normal election, the operational workflow is:

1. Voting server exports the ballot box to a removable media (e.g.: USB stick)
2. Voting server removable media is inserted by an election official into the Mixing server

3. Mixing server reads removable media and performs mixing
   - shuffles and re-encrypts votes without decrypting
4. Mixing server exports the mixed ballot box and mathematical proofs to removable media
5. Mixing server removable media is inserted by an election official into the Decryption server
6. Decryption server reads removable media and prompts the Electoral Board to load the decryption shares (the election private key components).
7. Decryption server decrypts ballot box and generates proofs of correct decryption
8. Decryption server exports ballot box and proofs to removable media

To represent this data flow visually:



Since the import and export operations are performed using removable media, it is necessary that an election official performs these steps manually. Therefore, the workflow is visible to anybody present for the process (other election officials, Electoral Board members, scrutineers and auditors). During the process all attendees have a checklist to follow the procedures being observed.

3.3.2 iVote required workflow to enable attack

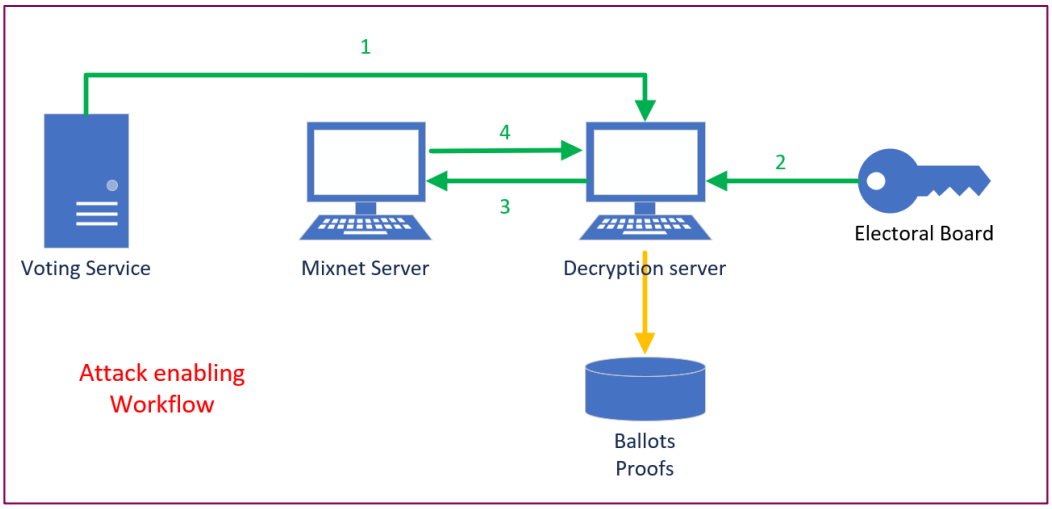
In order to allow the attack proposed in the Report the workflow needs to be changed to force the system to be in an adaptive scenario. That means that it is necessary that the decryption server evaluates the changes that need to be done in the target encrypted votes and give the instructions to the Mixnet to perform specific operations to hide the manipulation. Due to this requirement the process must start in the decryption server rather than the mixing server.

The following workflow is an example of one required in case of an attack. Note the changes in red. Furthermore, the red operations need to be executed using the attacker's manipulated software:

1. Voting server exports the ballot box to a removable media (e.g.: USB stick)
2. Voting server removable media is inserted by the attacker into the Decryption server in addition to randomness data from comprised voting devices and fingerprint of each encrypted vote.
3. Decryption server reads removable media and asks Electoral Board to introduce the decryption shares (the election private key components).
4. Decryption server identifies the specific target votes for which it has the randomness data, decrypts them, generates fake decryption proofs and calculates the new ciphertext of the vote that allows the cheating proof to be mathematically valid

5. Decryption server stores the original ballot box in removable media, along with the new ciphertext of the votes that will be attacked with the fingerprint of the original encrypted vote related to this new ciphertext.
6. Decryption server removable media is inserted by the attacker into the Mixing server
7. Mixing server loads the removable media, calculates a re-encryption permutation that generates the new ciphertext expected by the decryption server and shuffles and re-encrypts votes using this calculated permutation.
8. Mixing server exports mixed ballot box and proofs to removable media
9. Mixing server removable media is inserted by the attacker in Decryption server
10. Decryption server reads removable media and identifies the new ciphertexts generated in step 5 above.
11. Decryption server decrypts all the votes, generates valid decryption proofs for the votes that were not attacked and fake proofs for the votes that were attacked. *Note that attacked votes contain nonsense vote data (i.e.: invalid votes).*
12. Decryption server exports ballot box and proofs to removable media

To represent this data flow visually:



As mentioned earlier, the imports and exports are manual operations that must be performed by the attacker. Therefore, it will be clearly visible that the attacker is changing the workflow and this process is observed by Electoral Board members, the auditor, a number of election officials and any scrutineers in attendance.

Furthermore, the attacker goes back and forward between the separate decryption and mixing servers. This makes it even more clear to an observer that the process is not being followed, and greatly increases the risk of discovery of the attack whilst it is being performed.

3.4 Detecting the proposed attack

Detection of the attack is based on the observation and audit of the process. As part of the observation, auditors can take the following evidentiary items during the counting phase:

- The original ballot box exported from the iVote voting system.
- The shuffled ballot box and mixing proofs exported by the Mixing services.
- The decrypted votes and decryption proofs exported from the decryption server.

As in any election, it is important that there is independent scrutiny (Auditor, verifiers and scrutineers in the case of iVote) in addition to the Electoral Commission's procedures to ensure election officials do not have opportunities to influence the results.

Based on these assumptions, if auditors and reviewers of technical mathematical proofs are observing the workflow and taking evidence from the correct sources (e.g.: the shuffled ballot from the mixing server instead of the decryption server), the attack could be detected when auditing the decryption proofs as explained below.

3.4.1 Standard non-adaptive scenario

In a standard scenario (non-adaptive), where mixing and decryption servers cannot interact such as when they are in separate servers, in the case that the decryption server performs the attack, it will always be discovered when the prover (i.e.: the auditors) use the output of the mixing server (i.e.: the correct mixed votes).

In the case that the mixing output is manipulated following extraction from the mixing server, this manipulation will be detected by checking the digital signature of the mixing output (i.e.: data file comparison) or by crosschecking the mixing output with the evidence extracted from the observers.

In any of the cases, the attack will be always detected since it generates decrypted votes with nonsense data, as originally stated by the authors.

3.4.2 Complete manipulated adaptive scenario

This is the case for an environment where an attacker can control and manipulate several components of the iVote voting system to modify the ciphertext without being detected. This case also accommodates the scenario where an attacker implements a second mixing process within the decryption server and substitutes the Mixing data from that of the original mixing service.

The following items each propose a way in which the attack described above could be detected:

- Observers note the introduction of the election private key into any server prior to the mixing being performed
- Observers note the introduction of the election private key into to the mixing server
- Observers note the movement of data between systems outside the planned process
- Post-election review of ballot data reveals corrupted vote data
- Mathematical evaluation of the proof data emitted originally from primary data sources (e.g.: the Mixnet Server and the Decryption server) do not align

3.5 Mitigation measures

Even though the attack is always detected at the end of the decryption process, additional mitigation measures can be implemented to ensure any attack is detected when it happens. These measures are also recommended even in the case that the decryption proof is updated to support adaptive environments (i.e.: Strong Fiat-Shamir is implemented).

These are some mitigation measures, noting that all these were in place during the 2019 NSW election:

- Observers (a group comprising NSWEC staff, auditors, public attendees, political party representatives) are trained on the workflow for the decryption ceremony that takes place at the conclusion of the election.
- Steps are taken in a controlled environment
- Post-election review of proof artefacts
- The introduction of the election private key is a significant event requiring a number of trusted officials to operate in a co-ordinated manner, which would not proceed if it is not the process they are trained in or is documented for observers.

An additional mitigation, which Scytl is implementing in our software is:

- Visual feedback related to the outputs generated by different servers are shown on the screen allowing easier verification at the time of an event. As an example - showing on a system console the fingerprint of the exported data, so an observer can take a copy and a photograph. The fingerprint can be also shown on a server when data is imported to allow a crosscheck to ensure data is not manipulated. These measures do not substitute the verification of the digital signature of the data, but they do allow the performance of a more interactive audit of the election and facilitates the capture of evidence for later review.

3.6 Attack scalability

Another point to consider is the scalability of the attack. As discussed in section 3.2, the manipulation of a vote requires that the attacker learns the randomness used during the vote encryption for a specific vote. That means that having access to a ballot box is not enough to perform the attack, it is necessary for the attacker to spy on the voter (say through the use of a compromised voting device) when the vote is being cast to perform the attack. So to change 100 votes with associated fake proofs, it is necessary to compromise the devices of 100 voters. This limits the scalability of the attack, especially if it is compared with other scenarios in which having the control of a ballot box means having control over the whole set of votes inside without needing to spy on voters when they cast their votes (e.g.: postal voting).

4 Conclusion

Post-election review has shown that this vulnerability was not exploited, based on review of processes and procedures, and the lack of detection of nonsense votes – the existence of which may have indicated an attempt to attack the system in the manner described.

Scytl thanks the author of the Report for the analysis and the demonstration of the weakness of WeakFS as can be seen in Swiss Post voting system as already acknowledged. Based on the mitigation techniques in place in the NSW State Election 2019 Scytl is of the view that the decryption proof flaw identified in the Swiss Post voting system cannot be reproduced without being detected by any observer.

5 Addendum: Weak Fiat-Shamir vs Strong Fiat-Shamir

Scytl notes that the author of the Report correctly points out that the use of weak Fiat-Shamir (WeakFS) is a weakness in the system in use by Swiss Post in certain circumstances. This is correct, as the Mixing service and the Decryption service run on the same machine, making the use of WeakFS a vulnerability in the design, as has been accepted by Scytl in the past.

The use of WeakFS however does not create this vulnerability in the iVote system for the following significant reasons:

- Mixing service and Decryption service are physically isolated
- To exploit the vulnerability by simulating the Mixing and Decryption services on the same system results in evidence that is detectable in terms of detection of corrupted votes, existence of non-approved software, and deception of all observers who all have different electoral motives including any public in attendance
- Scytl has provided a patch to iVote which moves from the use of WeakFS to StrongFS, as provided to Swiss Post. This update brings a higher level of uniformity between the systems.
- The use of StrongFS removes the vulnerability completely in systems where the Mixing service and Decryption service are executed on the same server

Scytl has supplied a patch for the iVote system to support StrongFS rather than WeakFS.

6 Addendum 2: Protocol document update (Nov 2019)

Thanks to Associate Professor Teague in putting together the Report, Scytl will shortly update the protocol document (“NSWEC-SD-2 Voting protocol description”) to include further information regarding the use of the mixing proofs, to aid in understanding how they are used.

-- Document ends --