A Simple and Robust End-to-End Encryption Architecture for Anonymous and Secure Whistleblowing

Hariharan Jayakrishnan
Department of Computer Science and Engineering,
Amrita School of Engineering, Coimbatore,
Amrita Vishwa Vidyapeetham, India
Email: cb.en.u4cse15120@cb.students.amrita.edu

Ritwik Murali*
Department of Computer Science and Engineering,
Amrita School of Engineering, Coimbatore,
Amrita Vishwa Vidyapeetham, India
Email: m_ritwik@cb.amrita.edu

***Abstract*—Much has been mentioned about the importance of whistleblowing. While all organization are recommended to have a whistleblowing mechanism, there are very few completely software based platforms that assist in performing this task securely. The primary concerns are to ensure that the whistleblower remains anonymous and the disclosures are securely delivered to the competent authority, usually media organizations. In this paper we have analyzed the security of the state of the art software based whistleblowing platforms and related research, identified security issues and proposed a new architecture that satisfactorily ensures the requirements for anonymous and secure whistleblowing. We have verified the strength of our solution against the existing platforms and related research with the AVISPA (Automated Validation of Internet Security Protocols and Applications) tool. Our approach is practical, backed by cryptographic security and, because of its modularity, can be easily included into the current infrastructure of many whistleblowing platforms. The results show that our architecture is simple, robust and implements a complete end-to-end encryption strategy thus enabling secure and anonymous whistleblowing.**

***Index Terms*—Anonymity, Authentication, Confidentiality, Plausible Deniability, Privacy, Security Architecture, Usable Security, Whistleblowing, Whistleblowing Platforms.**

I. Introduction

Whistleblowing is the act of exposing any kind of alleged wrongdoing like violation of company policies/rules, law, regulation, or threat to public interest/national security, as well as fraud, and corruption [1]. It is now an essential tool within businesses or governments to protect their employees, citizens, customers, and their organisation as a whole from these activities[2]. Unfortunately, there are limitations to whistleblowing, since people fear repercussions. Therefore, this implies that the act of whistleblowing must ensure that the following fundamental requirements are met.

1) Anonymity of the Whistleblower
2) Confidentiality and Integrity of the Disclosures
3) Plausible Deniability of the Whistleblower
4) Authenticity of the Receiver

978-1-7281-3591-5/19/$31.00

To promote and encourage the culture of anonymous whistleblowing, there are many online whistleblowing platforms that enable whistleblowers to communicate anonymously with the media organizations. We classify the act of whistleblowing as non-software based and software based. The former includes physical means of communication like posting anonymous letters while the latter includes sharing of disclosures using digital means of communication which we further classify into two types.

1) Type 1: An organization directly accepts the submissions of the disclosures from the whistleblowers, analyses its veracity and authenticity with the help of news organizations and then decides to expose them to the public through either themselves or media organizations. Eg: Wikileaks [3]
2) Type 2: A software-based whistleblowing platform, which promises anonymity and data confidentiality, operates as a service to news and media organizations. The media organizations install an instance of these whistleblowing platforms and share the instance address publicly. The whistleblowers can then send their disclosures directly to the media organizations which then accept the disclosures, analyze its authenticity and then decides to expose them (or not) to the public. Eg: SecureDrop, GlobaLeaks

This paper focuses only on the Type 2 platforms as our interest is in the claims by the security platforms to provide all the requirements for whistleblowing. We believe that the architecture of the existing popular whistleblowing platforms are still vulnerable to an active adversary, controlling the server, who can easily intercept the disclosures before reaching the server, and hence access them directly in plaintext format. This creates a need for end-to-end encryption of the disclosures [4]. The disclosures can be encrypted in the client-side in the whistleblowers' browsers and then can be sent to the media organization's server.For software based whistleblowing platforms, the following two requirements must also be included.

1) The need for Utility (confidence that the disclosures

reach its intended recipient) of the platform.
2) The need for Usability of the platform.

Our solution involves the media organization server sending an authorized and verifiable script that performs client-side encryption and disappears from the client system without leaving a trace. This would help in ensuring anonymity of the whistleblower, allow room for plausible deniability on confrontation by the higher powers affected by the act and ensure the confidentiality and integrity of the disclosures. This also reduces the onus on the whistleblower who now does not need to be highly technologically savvy in order to use our proposed whistleblowing system thus proving usability. We also show that configuring our solution requires very little technical skills and ensures that the disclosures reach the intended recipient.

II. Related Work

A. Existing Whistleblowing Platforms

1) Case Study: SecureDrop: SecureDrop currently gets the submissions from their sources in plaintext and encrypts them server-side, which is not a good idea as discussed above. An attacker who compromised the SecureDrop server can easily read the submissions in plaintext, before getting encrypted and written to disk. Making the submissions to get encrypted client-side would solve the problem, but requires JavaScript to run in the client machine. In that case, a server compromised attacker can easily change the encryption key to his own before sending the pages to the source. SecureDrop's desire to convert to client-side infrastructure can be found here [5].

2) Case Study: GlobaLeaks: Similar in spirit to SecureDrop, GlobaLeaks too, currently depends on server-side encryption, and have a desire to implement client-side OpenPGP.js encryption, which can be found here [6].Our proposed approach can be tweaked a bit to be completely compatible with the GlobaLeaks' current infrastructure.

B. Related Research

There is very little work done on securing online whistleblowing platforms. Most of the work focuses on the ethics and implications rather than the technological aspects. In this section we perform a detailed analysis of the available work regarding the technical aspects of the whistleblowing platforms focusing on their security.

1) An End-to-End Encryption Scheme for SecureDrop [7]: In their architecture, Nater et al., state that the whistleblower is required to configure their browser extension during each visit to SecureDrop and manually enable and disable JavaScript for encryption. We believe this to be a negative approach. First off, this is not very user-friendly as it assumes the whistleblower has a fair bit of technical knowledge. Secondly, if the extension is wrongly configured or the JavaScript is enabled before the web page verification, a malicious script can get executed on the client side thereby exposing the whistleblower and thus making the scheme insecure. Also, the public encryption key, the web pages' signatures and the public verification key that is used for server authentication and encryption, are fetched from an unauthenticated server. This still does not solve the threats of a compromised server.

2) AdLeaks [8]: Here, Roth et.al., have implemented a Type 1 whistleblowing platform, similar in spirit to WikiLeaks. They use an advertisement network to mask the disclosures within an ad and route them to the desired servers with a client-side encryption approach.

One of the problems with this approach is that the whistleblowing platform cannot guarantee that the disclosure submissions have reached their intended recipient. This is because the act of submission is not under the control of the whistleblower as it is the AdLeaks platform that retrieves the disclosures from the whistleblowers' computer. Further, there are possibilities that the whistleblower will not encounter an AdLeaks ad at all. Moreover, even if the disclosures successfully reach the "decryptors", they aren't always recovered by them, For client-side encryption, AdLeaks uses "instrumented browsers". These are tweaks of generic browsers used to perform the JavaScript verification and disclosure encryption, using the AdLeaks encrypting tool. But, according to us, this is another threat to whistleblowers, since it can void their plausible deniability. The AdLeaks prime security seems to lie in its ad network design and implementation. However, the authors have not discussed the possibilities and after-effects of AdLeaks network compromises. If AdLeaks network is compromised by an active adversary, its architecture becomes completely insecure. Further, the adversary can easily spoof the signature of the installer and ads and also tweak the cryptographic keys. Also, the authors assume that all whistleblowers perform digital signature verification of the AdLeaks installer, before using it. However, normal users do not have such cryptographic knowledge and could miss that step. This could lead to the whistleblower being de-anonymized thanks to the use of a compromised version of the installer.

III. The Proposed Architecture

We would like to reiterate; every online whistleblowing platform has three parts.

1) The Client or the whistleblower who sends information.
2) The Whistleblowing platform that facilitates the disclosure submission.
3) The Journalist or news agency that receives the information and deems its value and newsworthiness.

Our primary focus is on parts 1 and 2. We assume that once the encrypted submissions reach the organizations' server, the members of the media organization such as editors or journalists can access them and decrypt them securely using security best practices.

Our proposed solution to this problem is to perform encryption client-side using JavaScript that is shipped from the server of the media organization and then send the disclosures anonymously from the client to the server using the tor hidden service. The submissions are encrypted with the organization servers' public key and this key is embedded inside the server response. A generic idea can be seen in Fig.1. Each module of the generic idea is described in detail in the rest

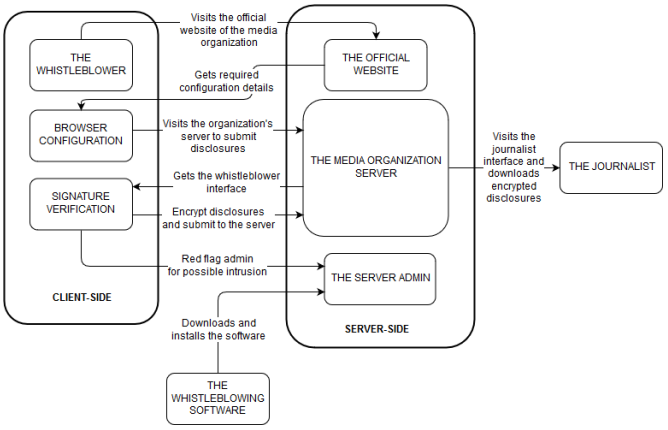

Fig. 1. The Overall Architecture

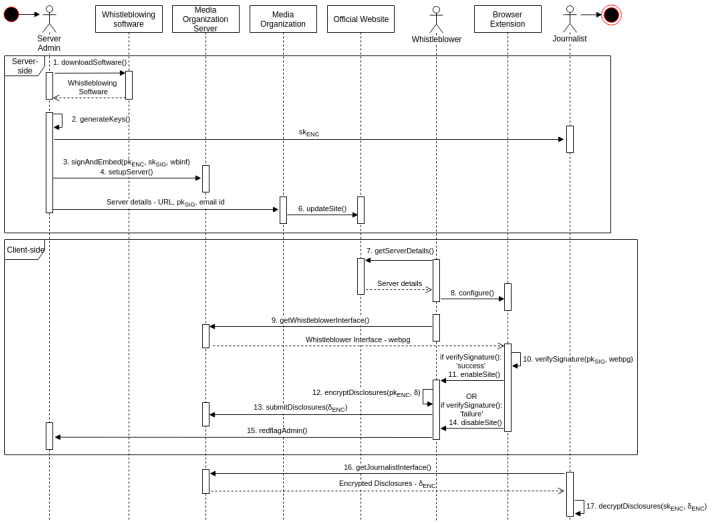

Fig. 2. Architecture Sequence Diagram

of the paper. On the outset, this seems a straightforward solution. However,our argument is that if that the server is compromised, performing client-side script execution imposes security threats like XSS and watering hole attacks, that could de-anonymize the whistleblower. To resolve this problem, our architecture states that only those scripts that are digitally signed by the administrator of the organization's server are executed on the client browser. This signature verification can be done by a simple browser extension. But as discussed in the previous sections, a special-purpose extension would void plausible deniability to the whistleblowers. Taking this into account, we have developed a general-purpose browser extension that checks the authenticity of the web pages. The extension can be configured with the media organizations' onion address and verification key by the whistleblower. All these information can be accessed by the whistleblower from the media organizations' official website as the data is publicly available. Further, this configuration requires only a rudimentary understanding of software thus ensuring usability. Any attempt by an adversary to modify the JavaScript or the server's public key would cause the signature verification to fail in the client-side and the extension would not allow any scripts to execute on the whistleblowers' browser. Further, if required, the admin can be notified anonymously about the possible intrusion. For this, we provide an optional feature of including the email address of the media organization in the extension. Our proposed architecture is very practical to implement, and can be easily augmented to current whistleblowing platforms because of its modularity. There are mainly 6 entities in our architecture namely,

1) The Whistleblowers - Anyone with the intent of providing a disclosure to specific media organizations.
2) The Whistleblowing platform/software - The software that media organizations install to anonymously accept the disclosures.
3) The Media or news organizations - Any organization, news agency or a public entity that wishes to accept anonymous disclosures.
4) The Server administrator - The server administrator of the organization that installs the whistleblowing software in the server and maintains it.
5) The Editors or Journalists - People belonging to the media organization that receive the encrypted disclosures and take it forward.
6) The Browser Extension - A generic tor browser extension that resides in the whistleblowers' browser which checks the authenticity of the web pages and allows script execution accordingly.

We have described the entities based on the functional sequence of our architecture Fig.2. We can use any existing semantically secure public-key encryption scheme, and a widely accepted digital signature scheme for establishing both confidentiality of the disclosures as well as authenticity of the server administrator.

Notations used:

- $G_{ENC}()$ - The Key Generation algorithm used for encryption and decryption of disclosures. It is a probabilistic algorithm that is invoked as $(pk_{ENC}, sk_{ENC}) \xleftarrow{R} G_{ENC}$.
- pk_{ENC} - The Public key used for encrypting disclosures.
- sk_{ENC} - The Secret key used for decrypting disclosures.
- $G_{SIG}()$ - The Key Generation algorithm used for signing and verifying the web pages. It is a probabilistic algorithm that is invoked as $(pk_{SIG}, sk_{SIG}) \xleftarrow{R} G_{SIG}$
- sk_{SIG} - The Secret key used for signing web pages.
- pk_{SIG} - The Public key used for verifying the web pages.
- $S()$ - The Signing algorithm. It is a probabilistic algorithm that is invoked as $\sigma \xleftarrow{R} S(sk_{SIG}, m)$, where σ is the signature.
- $V()$ - The Verification algorithm. It is a deterministic algorithm that is invoked as $\{accept, reject\} \leftarrow V(pk_{SIG}, m, \sigma)$
- δ - The Disclosures of the whistleblowers.
- $webpg_{STR}$ - The stripped web page after removing the

signature.

- $wbinf$ - The Whistleblower interface, which are the web pages that the whistleblower receives after requesting from the organization server.
- $html$ - The disclosures submission code in the whistleblower interface.
- js_{EMB} - The embedded JavaScript after appending the public key to it.
- $wbinf_{EMB}$ - The embedded whistleblower interface after constituting js_{EMB} and $html$.
- $wbinf_{SIG}$ - The signed whistleblower interface after appending the signature to $wbinf_{EMB}$.

A. The whistleblowing software

There are three modules here. At the very beginning of our software installation, the softwares' Signing and Embedding Module asks for the server's public key (pk_{ENC}) from the server admin, and embeds this key inside the JavaScript and HTML files that the whistleblowers access ($wbinf$). This public key is used for encrypting the disclosures in the client-side. Then, it asks for the organization's signing key (sk_{SIG}) from the admin and signs all the web pages with this key. The signing process of a web page is done by adding a comment at the top of the HTML file, that contains the PGP signature (σ) of the content of the <html>tag after it has been minified (Algorithm 1). The software provides an interface for the whistleblowers (The whistleblower Interface), that they can use to submit their disclosures (δ), and another interface for the media organization (The Journalist Interface), that the journalists can use to check for submissions and download them. When a whistleblower accesses the organization's onion address and submits a disclosure, the software accepts them, encrypts them in the client browser and sends it to the server. These encrypted disclosures (δ_{ENC}) get stored in the server's database for later access by the journalists.

Algorithm 1 Signing and Embedding

procedure signAndEmbed($pk_{ENC}, sk_{SIG}, wbinf$) :=
  $(html, js) \leftarrow wbinf$
  $js_{EMB} \leftarrow (pk_{ENC}||js)$
  $wbinf_{EMB} \leftarrow (html, js_{EMB})$
  $\sigma \xleftarrow{R} S(sk_{SIG}, wbinf_{EMB})$
  $wbinf_{SIG} \leftarrow (\sigma||wbinf_{EMB})$
  return $wbinf_{SIG}$
end procedure

B. The Server Admin

The admin generates the server's key pairs (Algorithm 2) and distributes the private key (sk_{ENC}) to the journalists, who store it on the workstations that they use for decryption of the encrypted disclosures. The admin then installs the whistleblowing software in the server and signs all the web pages that the whistleblowers access with the organization's signing key. The server runs a tor hidden service and resides inside the organization. The encrypted disclosures submitted by the whistleblowers are stored on a separate storage server which can be hosted with the onion address provided by the tor hidden service. Finally the server details (server's onion address, verification key (pk_{SIG}), admin's email address) are shared with the media organization which then publishes it on their official website.

Algorithm 2 Key Generation

procedure generateKeys() :=
  $(pk_{ENC}, sk_{ENC}) \xleftarrow{R} G_{ENC}$
  $(pk_{SIG}, sk_{SIG}) \xleftarrow{R} G_{SIG}$
  return $(pk_{ENC}, sk_{ENC}, pk_{SIG}, sk_{SIG})$
end procedure

C. The Media Organization

After setting up their whistleblowing server, the media organization publishes the server details in their official website, where the whistleblowers can access them easily and use them to configure their browser extension. Once they have this onion address, they can access the organization's whistleblowing service anonymously .

D. The Browser Extension

Our browser extension disables all the scripts from any site by default. When a user accesses a website, it checks if the website requested is added to its configuration, and if yes, fetches the corresponding verification key of the signer. It then strips the signature from the web page, if present, and verifies if the signature is correct with the previously fetched verification key (Algorithm 3). It allows scripts to execute, if and only if this web page authentication is successful. If unsuccessful, displays a bad page error to the whistleblower. Since the extension is generic and not specific-purpose, it provides plausible deniability for the whistleblowers. Configuring this extension is also straightforward and does not require specialized knowledge. Only the media organization server's onion address, verification key, admin's email address are required and these are already available from the media organizations official web page.

Algorithm 3 Signature Verification

procedure verifySignature($pk_{SIG}, webpg$) :=
  $(\sigma||webpg_{STR}) \leftarrow webpg$
  if $(V(pk_{SIG}, webpg_{STR}, \sigma) = reject)$ **then**
    return $failure$
  else
    return $success$
end procedure

E. The Whistleblowers

The whistleblowers install the Tor browser and the proposed browser extension from the add-on store. When they wish to submit a disclosure to a media organization, they find the server details from the media organizations' official website

and then configure the browser extension using these details. Only after this configuration step, they navigate to the organization's onion address and proceed to submit their disclosures. These disclosures are encrypted on the client-side within the browser itself. They are advised to use a Tails operating persistent volume to open the Tor browser and submit their disclosures, if they wish to completely erase all trails of their whistleblowing activity.

F. The Journalists

The organizations' journalists periodically access their interface and check for disclosure submissions. . If any submissions are found, they are downloaded and decrypted by the journalists with the server's private key (sk_{ENC}) that the organization's admin has previously provided. They too are advised to use a Tails operating persistent volume to open the Tor browser and check their interfaces to completely erase all trails of the decrypted disclosures from the system memory.

IV. Observations and Results

A. Security Assumptions

For successful implementation, there are a few prerequisites to be observed. We detail these as the security assumptions, but they can also be viewed as best practice guidelines for successful usage of our proposed architecture.

1) *Usage of Tor hidden service:*
   The media organizations are recommended to host the whistleblowing software only in a server that runs Tor hidden service.
2) *The media organization hosts the server:*
   The media organization hosts and manages a server and it is not managed by any third party.
3) *Usage of Tails operating persistent volume and Tor browser:*
   If the whistleblower and the journalist don't use Tails, a whistleblower's computer memory might have traces of information that exposes the whistleblowing activity also the journalist's computer memory might have traces of the decrypted disclosures. If Tor browser is not used, even a passive adversary that monitors the network could learn about the entities' identities. This voids anonymity of the whistleblower and the confidentiality of the disclosure.
4) *The official web server of the media organization is not compromised:*
   If this server is controlled by an adversary, he/she could spoof the onion address or the verification key thereby breaking the whistleblower's anonymity and the disclosure's confidentiality.
5) *The whistleblower enters data properly in the browser extension:*
   If not, the extension cannot perform a valid signature verification and this might cause a malicious script to execute in the client-side, voiding whistleblower's anonymity.
6) *All actors are trustworthy:*
   If not, the admin can easily provide a fake public key or sign the web pages with a fake signing key during installation, and the architecture becomes insecure.

B. Security Features

If the proposed architecture is correctly implemented and the assumptions mentioned above hold true, then this approach claims the following security features:

1) *Confidentiality.*
   Using end-to-end encryption ensures foolproof confidentiality of the disclosures submitted by the whistleblowers.
2) *No dependency on third party services.*
3) *User-friendly.*
   The web page verification and the encryption of disclosures are automated, so no user intervention is required and hence, very user-friendly.
4) *Complete Anonymity.*
   Using Tor hidden services completely anonymizes the server and the whistleblower, and prevents possible attacks in the onion routing protocol.
5) *No Tracability.*
   Using Tails completely erases the trails of any whistleblowing activity.
6) *Prevents unauthorizied script execution.*
   Authenticated script execution prevents running of malicious scripts in the client-side, preserving whistleblowers' anonymity, and server authenticity.
7) *Plausible Deniability.*
   Making the browser extension generic, enables plausible deniability for the whistleblowers, and makes it multipurpose.

C. Security Validation of the Proposed Architecture

As mentioned earlier, the goal of a software-based whistleblowing platform is to ensure the following.

1) Anonymity of the Whistleblower
2) Confidentiality and Integrity of the Disclosures
3) Plausible Deniability of the Whistleblower
4) Authenticity of the Receiver
5) Utility of the platform
6) Usability of the platform

To verify the correctness of our solution, we used the Automated Validation of Internet Security Protocols and Applications (AVISPA) tool. We have modeled our solution on AVISPA . $alice$ is the Media Organization Server and bob is the Whistleblower. The AVISPA goal is to verify authenticity of the receiver and confidentiality of the disclosures. The AVISPA analysis of our protocol was successful and it returned a summary as $SAFE$. This implies that the goals of authenticity and confidentiality (goals 2 and 4) have been met. The usage of the Tor hidden service from the server-side and the Tor browser from the client-side ensures that the anonymity of the whistleblower is maintained (goal 1) and the usage of our easily configurable general purpose browser extension satisfies

TABLE I
SECURITY ANALYSIS OF EXISTING SOLUTIONS

Technique/ organization	Vulnerabilities/ Issues	Attack Strategy/ Cause of Issues	Requirement Voided
Specific-purpose client-side software	Adversary's knowledge of possession or control	Inspecting whistleblowers workstation	Plausible deniability of the whistleblower, Receiver authenticity
Unauthenticated JavaScript for client-side encryption	Script injection	Server Compromise	Disclosure confidentiality, Whistleblower anonymity, Receiver authenticity
SecureDrop/ GlobaLeaks	Server-side Man-In-The-Middle	Server Compromise	Disclosure confidentiality, Receiver authenticity
E2EE for Secure Drop [7]	Failed browser configuration	Frequent Manual Configuration	Usability by whistleblower
	Script injection	Server Compromise	Disclosure confidentiality, Receiver authenticity, Whistleblower anonymity
AdLeaks [8]	Uncertain disclosure submission	Not encountering an ad, Decryptor probability	Utility of the whistleblowing platform
	Passive network sniffing	Network monitoring	Whistleblower anonymity
	Use of instrumented browsers & Software dissemination process	Inspecting whistleblower's workstation & activities	Plausible deniability of the whistleblower
	Spoofing of keys & signatures	Any network compromise	Disclosure confidentiality, Receiver authenticity, Whistleblower anonymity
	Failed signature verification	Usage by non-technical whistleblowers	Disclosure confidentiality, Receiver authenticity, Whistleblower anonymity

plausible deniability (goal 3). Overall due to the ease of use of our browser extension by the whistleblower, usability of the platform is achieved (goal 6). This proves that the utility of the platform is established as well (goal 5).

D. Comparison with Existing Solutions and Related Research

In order to verify the security of existing architectures and research, we have modeled the same in AVISPA as well. The $Intruder$ was the adversary controlling the server. In the SecureDrop and GlobaLeaks architectures, $bob.s1$ signifies the unencrypted disclosures of the whistleblower which could be accessed by the adversary. AVISPA returned the summary as $UNSAFE$ and reported a Secrecy attack on the architecture.

In the case of Unauthenticated Client Side Encryption and using E2EE for SecureDrop [7], $bob.s1\_ka$ is the legitimate whistleblower interface uploaded by the media organization which includes the correct public key and client-side encryption JavaScript. The adversary controlled web pages which includes the encryption public key and the client-side encryption JavaScript is denoted as $bob.var-s-1\_ka$. AVISPA returned the summary as $UNSAFE$ and reported an Authentication attack on the architecture.

A complete security analysis of the existing systems and related research have been summarized here (Table I) along with a comparative analysis of the whistleblowing requirements satisfied by all architectures (Table II).

TABLE II
COMPARISON OF WHISTLEBLOWING PLATFORMS AND TECHNIQUES

	Confidentiality of the disclosures	Anonymity of the whistleblower	Authenticity of the receiver	Plausible Deniability of the whistleblower	Usability by the whistleblower	Utility of the platform
Specific-purpose client-side software	Yes	Yes	No	No	Yes	Yes
Unauthenticated Javascript for client-side encryption	No	No	No	Yes	Yes	Yes
SecureDrop/ GlobaLeaks	No	Yes	No	Yes	Yes	Yes
E2EE for Secure Drop [7]	No	No	No	Yes	No	Yes
AdLeaks [8]	No	No	No	No	Yes	No
Our Architecture	Yes	Yes	Yes	Yes	Yes	Yes

V. CONCLUSION

A mechanism for whistleblowing is a necessary part of any organization. Since the act of whistleblowing itself is full of risks for the whistleblower, when designing a software solution that facilitates this process, the anonymity of the user and the confidentiality of the disclosures have to be ensured. End-to-end encryption strategies have become an obvious solution but, as described in this paper, most existing software based platforms and research in this domain, have not been able to satisfy all the fundamental requirements for a whistleblowing platform.

In this paper we have proposed a simple and robust end to end encryption architecture for secure and anonymous whistleblowing that ensures that whistleblowers also have the freedom to submit disclosures without fear of repercussions. We have also verified our architecture with the AVISPA tool and have found it to be "SAFE". We have also compared our architecture along with the existing whistleblowing platforms. Our proposed architecture is flexible enough to be customized and incorporated into existing systems thus providing a simple and improved software based solution for anonymous and secure whistleblowing.

REFERENCES

[1] W. Vandekerckhove, *Whistleblowing and organizational social responsibility: A global assessment*. Routledge, 2016.

[2] V. D. Avadhani, "Whistle blowing in banking sector-a swot analysis," *Indian Journal of Science and Technology*, vol. 9, no. 31, 2016.

[3] M. L. Sifry, *WikiLeaks and the Age of Transparency*. OR Books, 2011.

[4] A. Solad. (2018, Aug.) End-to-end-encryption: A reasonable balance for communication. Recorded Future Blog. USA. [Online]. Available: https://www.recordedfuture.com/end-to-end-encryption-communication/

[5] D. Auerbach. (2013, Oct.) Prototype encrypting data client-side with the system's public key. securedrop. [Online]. Available: https://github.com/freedomofpress/securedrop/issues/92

[6] GlobaLeaks. (2017, Apr.) Globaleaks encryption. GlobaLeaks. [Online]. Available: https://github.com/globaleaks

[7] P. W. Andres Nater, Erica Santana. (2018, May) An end-to-end encryption scheme for securedrop. MIT. [Online]. Available: https://courses.csail.mit.edu/6.857/2018/project/Nater-Santana-Wahl-SecureDrop.pdf

[8] V. Roth, B. Güldenring, E. Rieffel, S. Dietrich, and L. Ries, "A secure submission system for online whistleblowing platforms," in *International Conference on Financial Cryptography and Data Security*. Springer, 2013, pp. 354–361.