Implementation of RSA Blind Signatures in an Anonymous and Verifiable Digital Whistleblowing System

Luckman Fakhmanidris Arvasirri - 18223041
Department of Information System and Technology
School of Electrical Engineering and Informatics
Institut Teknologi Bandung, Jalan Ganesha 10, Bandung 40132, Indonesia
luckmanfaia@gmail.com, 18223041@std.stei.itb.ac.id

***Abstract*—A digital whistleblowing channel must protect the reporter while giving the recipient a reason to trust the submission. An open anonymous form accepts spam and unauthorised reports, while normal authentication exposes the reporter to the service that checks membership. This paper designs and implements a three-party protocol based on RSA blind signatures. The whistleblower hashes and blinds a report, the authority authenticates the requester and signs only the blinded value, and the receiver verifies the final signature without learning the reporter's identity. A Python prototype was evaluated through functional, transcript-ambiguity, tampering, forgery, and timing experiments. All 100 valid signing runs verified correctly. Repeated blinding produced 1,000 distinct values, and all 400 transcript-output pairings in a 20 x 20 matrix were algebraically consistent. Guessed signatures and modified reports were rejected. The prototype demonstrates the protocol mechanics but uses textbook RSA over a SHA-256 digest. A deployed system should use RSABSSA from RFC 9474 and an independently protected anonymous submission channel.**

***Index Terms*—blind signature, RSA, whistleblowing, anonymity, unlinkability, digital signature**

I. Introduction

Whistleblowing systems face a narrow but difficult trust problem. Organisations want reports from people with legitimate access to internal information, yet those people may stay silent if the reporting channel can expose them. Removing a name from a form does not solve the problem. Login records, network metadata, submission times, browser identifiers, and writing details can still reveal who contacted the service. Work on online whistleblowing therefore treats anonymity as a property of the complete system rather than a single interface setting [1], [2].

A fully open anonymous channel creates the opposite problem. The receiver cannot easily distinguish an authorised internal report from automated abuse or a submission created by an outsider. A normal login restores access control, but the authentication service then sees the member identity and the report in the same session. The desired property is more specific: confirm that the sender was eligible to receive an authorisation, then separate that check from the final report. Agrikola, Couteau, and Maier formalised a related goal as anonymous whistleblowing over authenticated channels [3].

Blind signatures provide that separation. Chaum introduced the idea so that a signer could authorise a value without seeing the message that would later carry the signature [4]. In the RSA variant, the user randomises the message representative before sending it to the signer. The signer authenticates the user and signs the randomised value. The user removes the random factor and obtains a standard RSA signature that another party can verify with the signer's public key.

This paper applies the mechanism to a three-party workflow. The whistleblower prepares and blinds the report. The authority verifies organisational membership and signs the blinded value. The receiver obtains the report and final signature through a separate anonymous channel. The design targets three claims: the authority should not learn the report during issuance, the issuance transcript should not identify the final report, and the receiver should be able to verify that the report was authorised and has not changed.

The contribution is practical rather than a new cryptographic primitive. It consists of a protocol design, a Python implementation of the four blind-signature operations, an experiment that checks every pairing between issuance transcripts and final signatures, and a performance comparison across RSA moduli from 1024 to 4096 bits. Each claim is tied to a measurable test. The code deliberately uses textbook RSA to expose the algebra, so it is not suitable for live reports. RFC 9474 specifies RSABSSA, a modern construction whose final output is verified as an RSA-PSS signature [9].

Section II summarises the mathematics and security properties. Section III presents the system design. Section IV describes the prototype, Section V defines the experiments, and Section VI reports the results. Section VII discusses security limits, followed by the conclusion in Section VIII.

II. Technical Background

A. Digital signatures and RSA

A digital signature binds a private key to a representation of a message. The signer uses the private key, while any verifier can use the corresponding public key. Practical signature schemes first hash the message. A later change to the report produces a different digest, so the existing signature no longer verifies.

The prototype uses SHA-256, but its direct mapping of a digest into the RSA domain is only a teaching device. Production RSA signatures require a defined encoding such as RSA-PSS [10], [11].

RSA starts with two large primes p and q. The modulus and Euler totient are

$$n = pq, \qquad \varphi(n) = (p-1)(q-1). \tag{1}$$

The public exponent e is chosen so that $\gcd(e, \varphi(n)) = 1$, and the private exponent is

$$d = e^{-1} \bmod \varphi(n). \tag{2}$$

The public key is (e, n) and the private key is (d, n). In the algebraic form used by the prototype, a representative m is signed and verified as

$$s = m^d \bmod n, \tag{3}$$

$$s^e \bmod n = m. \tag{4}$$

Rivest, Shamir, and Adleman introduced RSA for public-key encryption and digital signatures [5]. The equations explain the prototype, but raw RSA is deterministic and preserves multiplicative structure. It should not be confused with a secure deployed signature format.

B. Blind signatures and security goals

A blind-signature protocol has an issuance phase and a public verification phase. During issuance, the signer should not learn the message. After issuance, the user holds a signature that verifies in the normal way. Blindness concerns what the signer learns about the message. Unlinkability concerns whether the signer can connect a completed message-signature pair to one particular issuance session. Formal treatments also study unforgeability and the one-more forgery problem [6]-[8].

These properties do not provide network anonymity by themselves. A signer can be unable to read the report while still observing the requester's account, IP address, and exact request time. A receiver can verify a signature and still infer the author from a distinctive fact or writing style. Blind signing protects the link between authenticated eligibility and report content. The surrounding system must handle network metadata and operational separation.

C. RSA blind-signature equations

Let M be the report and let its representative be

$$m = \mathrm{OS2IP}(\text{SHA-256}(M)) \bmod n. \tag{5}$$

The whistleblower chooses a fresh random r with $\gcd(r, n) = 1$ and computes

$$m' = mr^e \bmod n. \tag{6}$$

The authority signs only the blinded value:

$$s' = (m')^d \bmod n. \tag{7}$$

The whistleblower removes the random factor with its modular inverse:

$$s = s'r^{-1} \bmod n. \tag{8}$$

Substitution gives $s' = m^d r^{ed} \bmod n$. For valid RSA group elements, the public and private exponents leave one factor of r, which is cancelled by r^{-1}. The receiver can then verify s with the normal public-key equation. The authority never receives M or r.

The same algebra explains transcript ambiguity. Given an issuance transcript (m'_i, s'_i) and a final pair (m_j, s_j), the candidate

$$r_{ij} = s'_i s_j^{-1} \bmod n \tag{9}$$

satisfies

$$m_j r_{ij}^e \equiv m'_i \pmod n. \tag{10}$$

for valid textbook RSA values. Every transcript can therefore be made consistent with every final signature. The protocol equations alone do not identify the original pairing. Section V tests this property across a complete matrix of sessions.

D. Security requirements for whistleblowing

The protocol is useful only if its goals are stated narrowly. Report confidentiality during issuance means that the authority sees a blinded representative, not the report text. It does not mean that the report is encrypted after it reaches the receiver. Unlinkability means that the issuance transcript does not identify one final signature from the RSA values alone. It does not cover timing, network, or content-based correlation. Verifiability means that the receiver can check an authority-issued signature over the report digest. It does not reveal the individual author.

Integrity follows from hashing the complete report before signing. Any edit changes the digest and causes verification to fail. Access control is handled at a different point: the authority checks whether the requester is an eligible member before it signs. This separation is the reason blind signatures fit the problem. The authority enforces eligibility without receiving the content, while the receiver validates the content without receiving the organisational identity.

The design also needs replay and version rules. A report package should contain a protocol version and key identifier so that the receiver knows which verification procedure and public key to use. A production system may also include a receiver-generated case nonce inside the hashed report when duplicate submission is undesirable. That nonce must be public and independent of the member identity, otherwise it can become a tracking value.

III. System Design

A. Entities and trust boundaries

The system has three roles. The whistleblower writes the report, hashes and blinds it, requests a signature, unblinds the response, verifies it locally, and submits the final package. The authority owns the RSA private key and a membership database. It authenticates the requester and signs the blinded representative, but it does not receive the report. The receiver accepts the report and final signature through a separate channel and verifies them with the authority's public key.

This division limits what either service can learn. The authority knows who requested an authorisation but not what the person later submitted. The receiver sees the report but receives no organisational identity. Shared analytics, precise timestamps, or combined server logs can recreate the link, so the two services should also be separated operationally.

B. Threat model and assumptions

The authority is honest-but-curious. It follows the protocol and membership policy, but it may store transcripts and try to match them to later reports. An external party may submit random signatures, modify a report, or use the receiver without first obtaining an authority signature. The receiver is assumed to apply verification correctly.

The private key remains secret, the random generator produces unpredictable values, and the whistleblower's device is not compromised before blinding. The final channel is assumed to hide network identity well enough for the experiment's scope. The report must not contain a name, employee number, document metadata, or another direct identifier. The model does not cover a malicious signing service, side-channel attacks, compromised endpoints, or a global observer.

C. Data format and protocol flow

The prototype signs the SHA-256 digest mapped into the RSA modulus. A deployed format should also hash a protocol label, version, and public-key identifier with the report. Domain separation prevents a signature issued for whistleblowing from being reused in an unrelated protocol. The final package contains the report, signature, protocol version, and key identifier, but no member identity.

The authority first generates an RSA key pair and publishes the public key. The whistleblower hashes the report, chooses r, and sends the blinded value through an authenticated session. The authority checks membership and returns s' if the request is permitted. The whistleblower unblinds the response, verifies it locally, closes the authenticated session, and sends the final package through the anonymous channel. The receiver recomputes the digest and accepts only if the signature verifies.

D. Design choices

Membership checks occur before blind signing. The authority may limit how many authorisations one member can request, but it should not insert a user-specific serial number into the signed message. Such a value would become a tracking tag in the final report. A production system could use one-time credentials or a quota recorded only at the authority, without embedding identity in the signed content.

E. Why the separation matters

A conventional authenticated form gives one service both identity and report content. The proposed design deliberately avoids that combination. The authority can answer one question only: was the requester allowed to obtain a signature? The receiver answers a different question: does this report carry a valid signature from the authority? Neither service needs the other service's data to complete its task.

This separation also limits the damage from a partial breach. Compromise of the receiver exposes report content but not the membership database. Compromise of the authority exposes authenticated issuance records but not the reports, provided the anonymous channel and client remain sound. A breach of both services, or access to shared network logs, removes much of this benefit. The design therefore treats organisational separation and log minimisation as part of the security boundary, not as optional administration.

IV. Prototype Implementation

A. Software structure

The prototype was written in Python 3.13.5 with PyCryptodome 3.23.0. The core module implements key generation, hashing, blinding, blind signing, unblinding, verification, and transcript checks. A simulation module models the three roles. An experiment module runs the tests and writes timing data to CSV, while a plotting module produces the performance chart.

The blinding function samples r until the modular inverse exists. Python's three-argument pow performs modular exponentiation without constructing the full power. The whistleblower keeps r locally and never sends it to either server. The receiver hashes the report again and accepts only when the public-key equation holds.

B. Core operations

```python
def blind(m, public_key):
    e, n = public_key
    while True:
        r = getRandomRange(2, n - 1)
        if gcd(r, n) == 1:
            return (m * pow(r, e, n)) % n, r

def sign_blinded(mb, private_key):
    d, n = private_key
    return pow(mb, d, n)

def unblind(sb, r, public_key):
    _, n = public_key
    return (sb * inverse(r, n)) % n

def verify(m, s, public_key):
    e, n = public_key
    return pow(s, e, n) == m
```

Listing 1. Core textbook RSA blind-signature operations.

The code uses raw RSA over a digest so that the intermediate values match the equations directly. It omits the RSA-PSS encoding, element validation, key checks, and message preparation required by RSABSSA. The implementation is suitable for reproducing the experiment, not for processing real reports.

C. Transcript consistency check

The implementation stores issuance transcripts only for the ambiguity experiment. For each transcript i and final output j, it computes r_{ij} from (9) and tests (10). A complete matrix is more informative than comparing one blinded value with one message because it checks whether any transcript stands out as the unique source of a final signature.

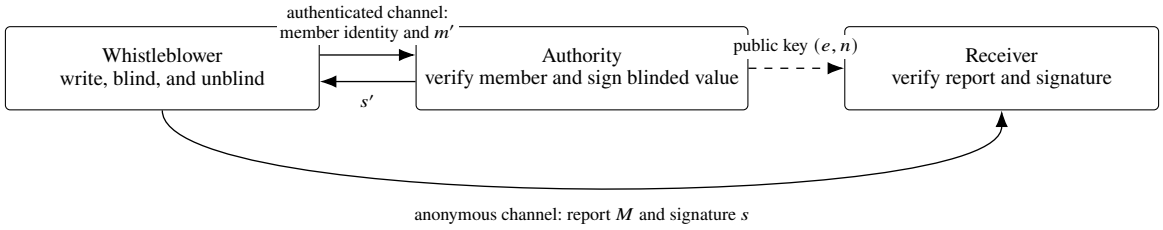


Fig. 1. Three-party architecture and separation of information. Source: author design.

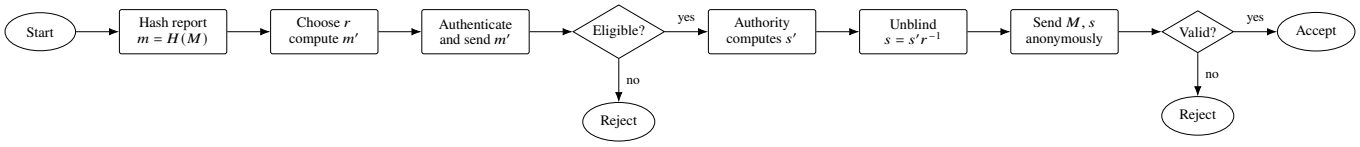


Fig. 2. Issuance and verification flow. Source: author design.

D. State, serialisation, and logging

The whistleblower stores the random factor only for the duration of one issuance session. Reusing the same factor would create an avoidable relation between requests, while losing it before unblinding would make the signature unusable. The prototype keeps the value in memory and discards it after local verification. A browser implementation would need similar care because persistent storage, crash recovery, or telemetry can accidentally retain sensitive state.

The experimental code represents RSA values as integers, but a real client must define a canonical byte encoding. The same report must produce the same digest on every platform. Text normalisation, line endings, attachment order, and file metadata can otherwise change the value being signed. A practical package should sign a structured representation that fixes field names, lengths, protocol version, and attachment hashes before SHA-256 is applied.

The authority log is intentionally small. It records enough information to reproduce the transcript-ambiguity experiment, but a deployed service should first ask whether those records are needed at all. Combining the member identity, exact time, network address, and blinded value in one long-lived log creates a correlation dataset. Operational monitoring can often use aggregate counts and error rates without retaining the full issuance event.

V. Experimental Method

A. Functional and negative tests

The functional test processed 100 different reports. Each report was hashed, blinded, signed, unblinded, and verified. The final signature also had to equal the direct textbook RSA signature $m^d \bmod n$. A second test blinded one fixed report 1,000 times with a new random factor and counted the number of distinct blinded values.

The ambiguity test generated 20 issuance transcripts and 20 final report-signature pairs, then evaluated all 400 pairings. The negative tests used 100 guessed signatures and 100 reports modified after signing. A guessed signature or a changed report counted as a failure only if the receiver accepted it.

B. Performance test

Blinding, signing, unblinding, and verification were timed with a high-resolution monotonic clock. Each operation was repeated 30 times for 1024, 2048, 3072, and 4096-bit RSA moduli. Key generation was excluded because an authority would reuse a key across many reports. RSA 1024 was included only as a performance baseline. It is not appropriate for a new deployment.

C. Metrics and reproducibility

The functional metrics were binary: signature verification, equality with the direct textbook RSA result, acceptance of a guessed signature, and acceptance of a modified report. The blinding test counted unique values, while the ambiguity test counted consistent transcript-output pairs. Timing results report the arithmetic mean and standard deviation in milliseconds. These measurements separate protocol correctness from performance and avoid treating a failed forgery guess as a quantitative security estimate.

Each experiment generated a fresh RSA key for its selected modulus and used independent random values for every issuance. The report strings were generated deterministically so that a second run exercises the same message set while still obtaining new blinded values. The CSV output records the key size, operation, repetition number, and measured duration. This makes the chart reproducible and allows another environment to replace the timing table without changing the rest of the paper.

The test environment was a Linux x86-64 container using Python 3.13.5 and PyCryptodome 3.23.0. Virtualised CPU allocation and background load can affect absolute timing, so the results should be read as a comparison among operations and key sizes rather than a hardware benchmark. The functional and algebraic results do not depend on that timing environment.

TABLE I
FUNCTIONAL AND NEGATIVE-TEST RESULTS

Test	Valid	Total
Blind-path verification	100	100
Equal to direct RSA	100	100
Distinct blinded values	1000	1000
Consistent transcript pairs	400	400
Guessed signatures accepted	0	100
Modified reports accepted	0	100

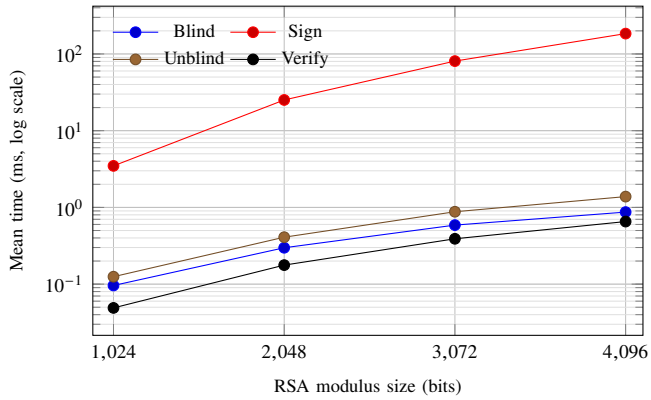


Fig. 3. Mean operation time on a logarithmic scale. Source: author experiment.

VI. Results and Discussion

A. Correctness, ambiguity, and tampering

Table I summarises the functional results. All 100 blind-path signatures verified and matched the direct textbook RSA result. This confirms that the random factor was removed correctly and that the four core operations were implemented consistently.

The same report produced 1,000 distinct blinded representatives. With a fresh random factor and a modulus of this size, collisions are not expected at that sample count. The useful result is that the implementation did not reuse r because of a state or random-number error. This test supports the implementation, but it is not a formal proof of blindness.

The ambiguity matrix contained 400 consistent pairings out of 400. Every issuance transcript could be paired algebraically with every final signature through a candidate r_{ij}. An authority that has only the RSA values does not obtain a unique match. This is the strongest experiment in the paper because it tests the full relation between stored transcripts and all final outputs, not just visual differences between large integers.

No guessed signature was accepted, and every report changed after signing was rejected. In the second case, the receiver computed a new digest while the signature still represented the original report. These are useful negative tests, but 100 failed guesses do not establish cryptographic unforgeability. That claim depends on the construction and its security model [8].

B. Performance

Signing was the most expensive operation because it used the large private exponent. Verification was much faster because the generated keys used the common public exponent 65537. Blinding and unblinding remained below 1.4 ms even for the 4096-bit modulus. The signing mean rose from 25.140 ms at 2048 bits to 80.725 ms at 3072 bits and 184.147 ms at 4096 bits.

A whistleblowing service is usually a low-frequency system, so these values are unlikely to be the main bottleneck. Authentication, the anonymous network, secure storage, and human review will usually cost more. The 3072-bit result is a useful reference because NIST associates that modulus with roughly 128 bits of security strength [12]. The 4096-bit setting more than doubled the signing time in this environment, so a larger modulus should be chosen only when the security policy justifies it.

C. Interpretation of the three main claims

The blindness claim is supported at two levels. The implementation uses a new invertible random factor for every request, and the repeated-blinding test confirms that the resulting representatives change across sessions. The algebra explains why the authority cannot remove that factor without the user's secret r. The experiment does not model side channels, biased randomness, or a malicious client, so it supports the implemented mechanism rather than a universal claim about every deployment.

The unlinkability claim is addressed by the complete 20×20 matrix. A simple demonstration that m' differs from m says little about later linkage. The matrix instead asks whether a stored transcript can be ruled out for a particular final signature. Because all 400 pairings satisfy the consistency equation, the RSA transcript does not select one final output. This conclusion holds only for the protocol values. A timestamp, network route, or unique detail in the report can still reveal the true relation.

The verifiability claim is the most direct. Every valid output passed the public-key equation, while each modified report failed. The receiver therefore obtains evidence that the authority signed the digest represented by the final report. That evidence is an authorisation statement, not proof that the report is true and not proof of who wrote it. Investigative procedures remain responsible for assessing the report's substance.

D. Meaning of the results

The experiments support three limited claims. First, the implementation applies fresh random blinding and hides the report from the authority during the issuance exchange. Second, the transcript matrix shows that the textbook RSA values do not provide a unique link to a final signature. Third, valid reports verify while modified reports fail. None of these observations removes the need for formal security arguments or a safe production construction.

VII. Security Analysis and Limitations

A. Blindness and metadata correlation

During issuance, the authority receives the member identity and m', but not M, m, or r. With a fresh invertible r, the factor r^e randomises the representative. The authority cannot test a

TABLE II
RSA BLIND-SIGNATURE OPERATION TIME, 30 REPETITIONS PER KEY SIZE (MS)

Bits	Blind		Sign		Unblind		Verify	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
1024	0.096	0.015	3.479	0.045	0.125	0.009	0.049	0.002
2048	0.298	0.063	25.140	0.308	0.409	0.021	0.177	0.005
3072	0.588	0.091	80.725	4.359	0.878	0.129	0.390	0.035
4096	0.869	0.088	184.147	1.299	1.383	0.109	0.654	0.017

candidate report by comparing it directly with m'. Blindness can still fail if the client leaks r, reuses it, or uses weak randomness.

System-level unlinkability also depends on time and network separation. If a member requests a signature at 10:01 and a report appears at 10:02 on an observable path, the timing may be enough to connect them. Useful mitigations include an anonymous network, delayed or batched submission, separate service operators, and short retention for logs that are not needed. Report content can also identify its author. Cryptography cannot remove clues created by unique facts or document metadata.

B. Verifiability and abuse

The receiver verifies that the signature matches the report digest and the authority's public key. It learns that the authority issued an authorisation, not which employee wrote the report. The authority cannot inspect report content when deciding whether to sign, so its policy is limited to membership, quotas, and other administrative conditions.

A valid member can request several signatures and submit several reports. Basic blind signatures do not prevent double reporting. Issuance limits can reduce abuse, but they only control how many authorisations are created. The receiver must also validate report size and format before expensive processing. RSA verification is relatively cheap, yet a flood of invalid packages can still consume network and storage resources.

C. Textbook RSA and RFC 9474

The largest limitation is the use of textbook RSA on a SHA-256 digest. Raw RSA is deterministic and has multiplicative structure. RFC 9474 defines RSABSSA with RSA-PSS encoding, message preparation, element checks, and parameter rules that are absent from this prototype [9]. Moving to that standard is not a one-line change. A production implementation should use a reviewed library and official test vectors rather than translating the equations directly into application code.

D. Key and endpoint compromise

If the authority's private key is stolen, an attacker can create authorised-looking reports without passing membership checks. The key should be isolated in a hardware security module or a dedicated signing service, with a key identifier in each final package. Compromise of the whistleblower's device exposes the report before blinding, while compromise of the receiver exposes it after delivery. Blind signatures do not protect an endpoint that is already controlled by an attacker.

E. Repeated use, replay, and denial of service

A valid member can obtain several signatures and submit several reports. Basic blind signatures do not provide one-report-per-person semantics because the final receiver cannot link a signature to the member who requested it. The authority can enforce an issuance quota, but it can only count authorisations, not prove how a particular authorisation was used. Stronger designs may use privacy-preserving one-time credentials or anonymous tokens, although those mechanisms add complexity beyond the scope of this prototype.

Replay is a separate issue. A valid report package can be copied and submitted again unless the receiver records a report identifier or signed nonce. Deduplication by the report hash is simple but may also merge two independently submitted identical reports. A receiver-generated public nonce offers clearer semantics, provided it is included in the signed digest and is not tied to the authenticated member.

The receiver may also face a flood of malformed packages. Format checks, report-size limits, attachment quotas, and queueing should occur before expensive storage or review. Public-key verification is cheap compared with private-key signing, but it is not free at large volume. The anonymous network may need its own rate controls that do not create a persistent identifier for the reporter.

F. Deployment profile

A safer deployment would replace the arithmetic core with an audited RSABSSA implementation, isolate the private key, publish versioned public keys, and keep the authority and receiver under separate access controls. Issuance requests could be processed in time windows rather than immediately, reducing simple timing correlation. The final submission channel should remove transport identifiers and document metadata before the receiver stores the report.

Audit records should answer operational questions without creating an identity-to-report map. Useful examples include the number of signatures issued per day, failed membership checks, key-health events, and receiver verification failures. Exact member identifiers, blinded values, and submission timestamps should not be retained together unless a documented requirement outweighs the privacy risk.

VIII. Conclusion

This paper designed and implemented a three-party whistleblowing protocol that separates membership checking from report intake. The whistleblower blinds the report digest, the authority signs the blinded value after authenticating the member, and the receiver verifies the final signature without receiving the member identity.

All 100 valid runs verified correctly. Repeated blinding produced 1,000 distinct values, and every one of the 400 transcript-output pairings was algebraically consistent. Guessed signatures and modified reports were rejected. Performance measurements showed that private-key signing dominates the cost, reaching about 81 ms for RSA 3072 in the test environment.

The results show that the proposed workflow is implementable and that its algebraic behaviour matches the intended blindness, unlinkability, and verification properties. They do not prove production security. A real deployment needs RSABSSA, a well-tested anonymous channel, isolated keys, careful log policy, and controls for metadata and endpoint compromise.

Acknowledgment

All praise and gratitude are due to Allah SWT, whose grace and guidance made it possible for the author to complete this paper. The author sincerely thanks Dr. Ir. Rinaldi Munir, M.T., lecturer of II4021 Cryptography, for the knowledge, direction, and support provided throughout the course. The author is also deeply grateful to their parents as well as their friends for their constant encouragement

References

[1] V. Roth, B. Guldenring, E. Rieffel, S. Dietrich, and L. Ries, "A secure submission system for online whistleblowing platforms," in *Financial Cryptography and Data Security*, LNCS 7859, Springer, 2013, pp. 354–361, doi: 10.1007/978-3-642-39884-1_30.

[2] B. Berendt and S. Schiffner, "Whistleblower protection in the digital age: Why anonymous is not enough. From technology to a wider view of governance," *International Review of Information Ethics*, vol. 31, no. 1, pp. 1–16, 2022, doi: 10.29173/irie479.

[3] T. Agrikola, G. Couteau, and S. Maier, "Anonymous whistleblowing over authenticated channels," in *Theory of Cryptography*, TCC 2022, Part II, LNCS 13748, Springer, 2023, pp. 685–714, doi: 10.1007/978-3-031-22365-5_24.

[4] D. Chaum, "Blind signatures for untraceable payments," in *Advances in Cryptology: Proceedings of CRYPTO 82*, Springer, 1983, pp. 199–203, doi: 10.1007/978-1-4757-0602-4_18.

[5] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978, doi: 10.1145/359340.359342.

[6] A. Juels, M. Luby, and R. Ostrovsky, "Security of blind digital signatures," in *Advances in Cryptology–CRYPTO 97*, LNCS 1294, Springer, 1997, pp. 150–164, doi: 10.1007/BFb0052233.

[7] D. Pointcheval and J. Stern, "Provably secure blind signature schemes," in *Advances in Cryptology–ASIACRYPT 96*, LNCS 1163, Springer, 1996, pp. 252–265, doi: 10.1007/BFb0034852.

[8] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko, "The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme," *Journal of Cryptology*, vol. 16, no. 3, pp. 185–215, 2003, doi: 10.1007/s00145-002-0120-1.

[9] F. Denis, F. Jacobs, and C. A. Wood, "RSA blind signatures," RFC 9474, Oct. 2023.

[10] National Institute of Standards and Technology, "Secure Hash Standard," FIPS PUB 180-4, Aug. 2015.

[11] K. Moriarty, B. Kaliski, J. Jonsson, and A. Rusch, "PKCS #1: RSA cryptography specifications version 2.2," RFC 8017, Nov. 2016.

[12] E. Barker, "Recommendation for key management: Part 1–General," NIST SP 800-57 Part 1 Rev. 5, May 2020.

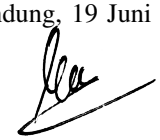
Source Code

The implementation and source code for this paper are accessible through this github link:
https://github.com/Arva05/blind-signature

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran atau terjemahan dari makalah orang lain, dan bukan plagiasi

Bandung, 19 Juni 2026



Luckman Fakhmanidris Arvasirri
18223041