International Journal of Computer Applications (0975 – 8887)
Volume 187 – No.44, September 2025

ZkVault: A Privacy-Preserving Smart Contract Framework for Confidential Transactions and Data Feeds

Jitendra Sharma
Research Scholar
Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore

Jigyasu Dubey, PhD
Head and Supervisor
Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore

ABSTRACT
Smart contracts are gradually replacing traditional text-based contracts, as they clearly demonstrate the power of blockchain technology in developing digital contracting that promises decentralized and contract execution without third-party intervention. The open nature of the public blockchains, however, reveals transactional information and smart contract logic, which can present a sizable privacy threat in some highly regulated industries, like healthcare, finance, and identity management. In order to alleviate these constraints, we propose a modular privacy-preserving framework called ZkVault that separates the computation problem, proof generation, and verification into three separate components, to be more flexible and extendable. ZkVault can easily replace secure oracle modules in order to read privacy-preserving off-chain data feeds that allow dynamic contract interaction with external data. Constructed out of zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARKs) and Pedersen commitments, ZkVault ensures: (1) data confidentiality hiding the inputs and intermediate computations; (2) verifiable computation which can be proven to third parties that no contract stages have been corrupted, without revealing any private data; and (3) scalability which off-loads most of the computation off-chain to reduce blockchain overhead. According to experiments that demonstrate that ZkVault can achieve a 40-60 % gas savings over legacy on-chain mechanisms, proof generation imposes a relatively small overhead of ~4s per transaction, which in practice is reasonable. These findings show that ZkVault can provide very robust privacy guarantees and implementation efficiency without sacrificing auditability, and as such, is a strong candidate for next-generation privacy-preserving dApps.

Keywords
Zero-Knowledge Proofs, zk-SNARKs, Groth16, Pedersen Commitments, Oracle Security, DeFi Privacy, Smart Contracts, Blockchain Confidentiality.

1. INTRODUCTION
Blockchain technology has revolutionized the world of digital technology to provide tamper-protected distributed ledgers that do not require trusted central authorities [1, 2]. Among the most impressive developments undertaken are smart contracts, self-executing program codes that facilitate agreements whose terms are executed in a transparent and immutable manner through blockchain consensus [3]. The downside of this transparency is the loss of privacy: transaction values, contract inputs, and executed logic are all publicly visible in a public blockchain like Ethereum, which is not acceptable in privacy-sensitive use cases, such as healthcare, decentralized finance (DeFi), supply-chain management, and identity management [4, 5]. Public blockchains are prone to disclosure of sensitive data (sender/receiver addresses, function parameters, business rules, and data provided by an oracle): revealing financial position, health records, or trade secrets [6, 7]. Besides on-chain privacy issues, smart contracts are often dependent on off-chain data sources via oracles, which can be compromised by data leakage/manipulation and malicious ordering to threaten accuracy and fairness [8, 9]. To remedy these privacy concerns, zero-knowledge proofs (zk-SNARKs) and subsequent technology, including Zcash, Aztec, Nightfall, and Mimblewimble, were discovered to enforce the confidentiality of transactions [10, 11, 12, 13], whereas schemes like DECO and Town Crier ensure oracle integrity [14, 15]. However, the majority of these frameworks have nonetheless considered either oracle correctness or transaction privacy at the cost of performance, i.e., high gas cost, large proof size, or slow generation of proofs [16, 17]. The requirement to comply with data privacy laws, such as GDPR, HIPAA, and financial regulations, also requires privacy assurance without compromising auditability and verifiability, a space where existing solutions fail [18, 19]. We address these limitations by creating ZkVault, a privacy-preserving, end-to-end smart contract framework, which combines confidential transactions with privacy-averse oracle feeds into a unified system. ZkVault uses zk-SNARKs and Pedersen commitments to (1) hide inputs and intermediate states of contract execution, (2) have those intermediate states publicly verifiable through a succinct proof, and (3) off-chain proof generation to achieve scalability [1, 17]. Experiments show that ZkVault uses 40-60% less gas than conventional transparent execution with acceptable proof generation times (~4s) [1, 16].

The major objectives of the current study have been to address the following two questions:

1. To protect the privacy of smart-contract transactions, without compromising auditability or soundness.

2. To deliver confidentiality of off-chain data feeds (oracles) by feeding encrypted, attestable privacy-preserving data to on-chain logic.

The current implementation of ZkVault carries out these two goals, but the architecture is also sufficiently modular so that one can, in a sense, explore these additional two goals as extensions as opposed to current contributions:

3. Trusted-oracle reputation scoring attempted to perform these tasks in a cryptographically verifiable way, called reputation systems.

4. Mitigation of transaction-ordering dependence (TOD) and frontrunning using commitment schemes and verifiable-delay cryptography.

ZkVault proposes a scalable application of zero-knowledge cryptography as a part of smart contract execution and oracle pipelines, thus providing an efficient and expandable framework of next-generation privacy-preserving decentralized applications.

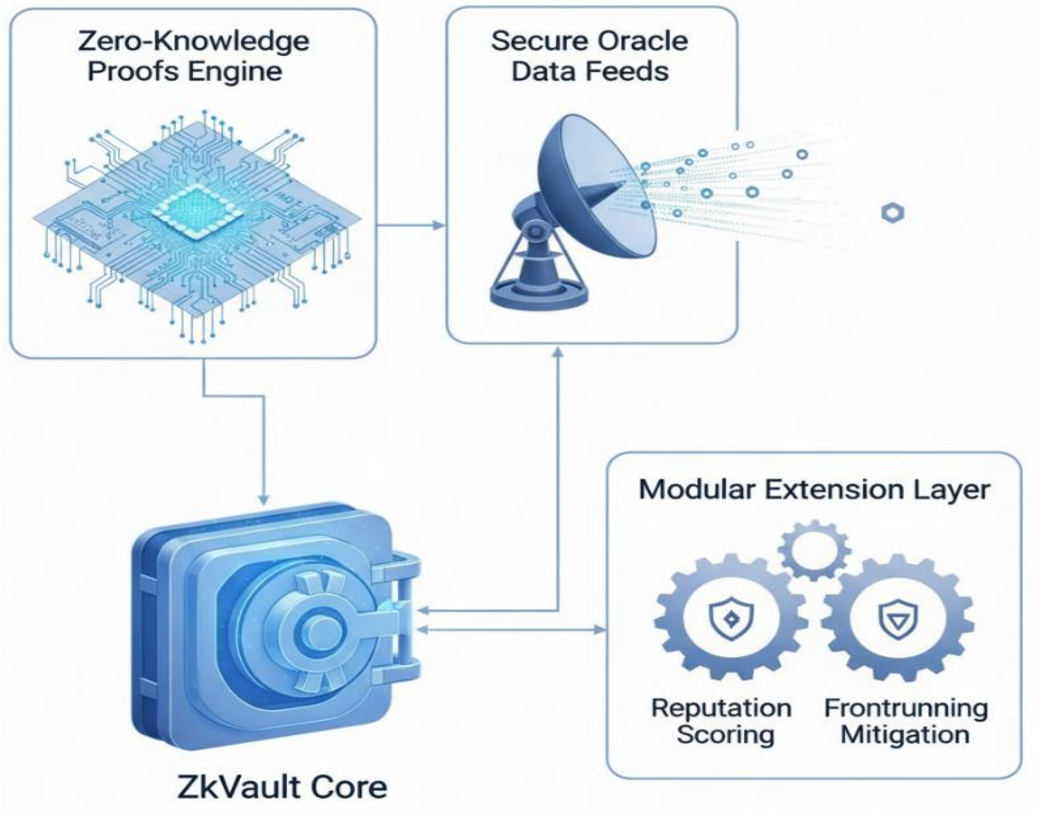


Fig 1: ZkVault Modular Architecture & Oracle Integration

2. PROBLEM STATEMENT

Even though smart contracts can potentially revolutionize the automation of trustless transactions, they're not being inherently privacy-aware may translate to serious downsides in privacy-sensitive sectors of finance, healthcare, and corporate information handling [1, 3, 4]. Smart contract information (such as input/output parameters of functions and internal state) is openly available on the blockchain, thus leading to essentially no transaction privacy [3, 5, 6]. This transparency may reveal sensitive data and may cause competitive drawbacks and subject participants to bad actors [6, 7]. Also, most smart contracts use external data feeds (oracles) to interact with the real world [5, 7]. Nonetheless, there are two significant threats of such oracles, i.e., data leakage and trust warranties [6 - 8]. When the external data is committed to the chain, it becomes publicly readable, which is of concern to confidential applications like pricing, medical, or sensitive operations -user metrics [4, 5]. There is also a chance of manipulation, inaccuracy, and breached contract logic due to a lack of credible mechanisms to verify the credibility and source of oracle data [6, 7, 18]. This study particularly looks at these two important issues by concentrating on:

- How to create a mechanism of personal transaction in smart contracts with Zero-Knowledge Proof.
- Crafting a solution that would allow confidential off-chain data feeds to be safely added to smart contracts.

The presented solution, ZkVault, may find its application in the existing critical gap between the blockchain transparency and decentralization on the one hand and data privacy on the other.

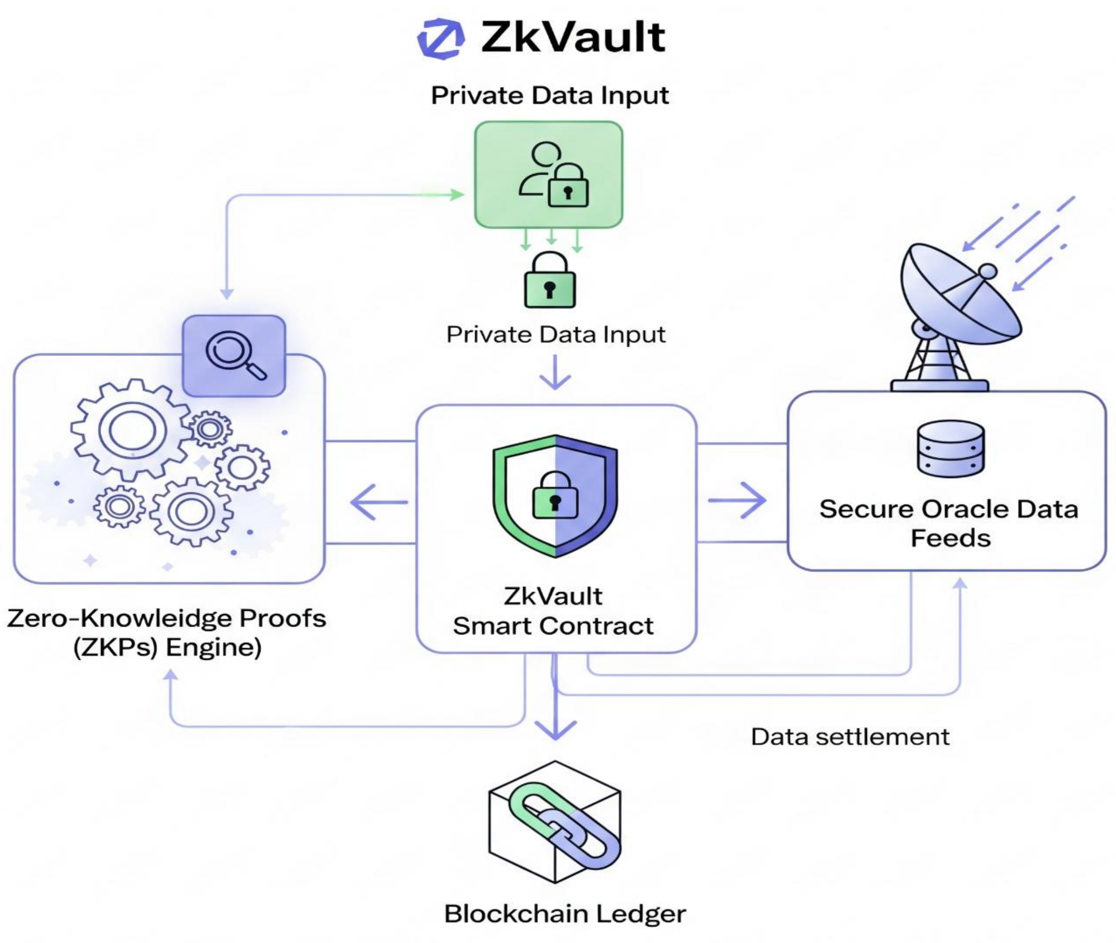


Fig 2: zkVault

3. LITERATURE REVIEW

Such a need to protect the privacy of the blockchain-based smart contracts has triggered the search for numerous cryptographic means to shield the information about transactions and protect the external data sources. In this section, relevant work is reviewed in two main areas, namely, confidentiality of transactions and secure oracle data feeds, which, respectively, directly address the aims of the first two objectives of this study.

3.1 Privacy in Smart Contract Transactions

There exists inherent transparency in smart contracts; every step of actions, reading, and writing is observable on the blockchain. In order to alleviate that, Zero-Knowledge Proofs (ZKPs), particularly zk-SNARKs (Succinct Non-Interactive Arguments of Knowledge) have found widespread use. ZKPs enable one party to demonstrate that data is correct, and this demonstration does not require disclosure of information. Shielded transactions such as Zcash are based on this principle since they allow values and addresses to be secret, but not the transaction, as the lender and borrower can prove [1]. Ernst & Young is developing zk-SNARKs projects such as Nightfall and Aztec Protocol for Ethereum to allow zk-SNARKs-based asset transfers. Nonetheless, these deployments can be characterized by high computational overhead, excessive proof times, and gas-intensive processes, and thus are not suitable when frequent use of smart contracts is intended [2][3]. Others like Mimblewimble and Confidential Transactions (CTs) are based on Pedersen Commitments and range proofs. The Grin and Beam make use of Mimblewimble, which uses an approach to scalability and confidentiality that does not involve address data and bundles transactions together [4]. As productive, these methods do not fit the Ethereum Virtual Machine (EVM), which hinders their wider application to Solidity-based contracts. Even with advances, all present systems tend to prioritize privacy or composability, but not both. This supports the necessity of a framework such as ZkVault, which allows finding a balance between privacy, verifiability, and deployability in EVM settings.

3.2 Privacy and Trust in Oracle Data Feeds

Blockchain oracles retrieve the information on the chain in smart contracts. Examples of outstanding oracle networks are Chainlink, Band Protocol, and API3, which have assisted in decentralization of oracles and data access [5]. Nevertheless, such solutions do not ensure the confidentiality of the data after passing into the blockchain. Oracle data submitted is immediately made on-chain and permanent. Examples of research work toward better oracle security and privacy are Town Crier, which uses Trusted Execution Environments (TEE) such as Intel SGX to safely load HTTPS-protected data to serve smart contracts [6]. In a similar way, the DECO, which is developed by Chainlink Labs and Cornell University, leverages zero-knowledge techniques to prove the facts relating to the web data that lies in the privacy [7]. Although it sounds promising, using software and hardware enclaves creates centralization risks and vulnerabilities.

In addition, although cryptographic signatures ensure the correct identity of oracles, they do not protect against leakage, and existing solutions are thus inadequate for applications such as privacy-preserving auctions, financial derivatives, or confidential clinical reporting.

3.3 Gaps and Research Need

Privacy-related practices that are promising in smart contracts and oracle design are found in the literature. However, the current frameworks only support one part of the privacy of transactions or trust of oracles. There are a few systems that have a unified solution of both, and a solution that can fit EVM and scale to a real deployment. The table 1 summarize existing privacy preserving smart contract and oracle solution

This gap drives towards the creation of a ZkVault modular and extendable framework that:

- Facilitates the confidentiality of transactions in smart contracts that are zk-SNARK-based.
- Endorses Oracle data feeds that are secure, encrypted, and verifiable.
- Sets the basis of future extensions of oracle scoring and transaction-ordering defense.

ZkVault can thereby effectively solve the two problems of transaction and oracle privacy in one fell swoop and provide a realistic way to fully confidential decentralized apps.

Table 1. Comparison of Existing Privacy-Preserving Smart Contract and Oracle Solutions

Framework / Protocol	**Privacy Focus**	**EVM Compatible**	**Oracle Privacy**	**Gas Cost & Proof Time**	**Limitations**
Zcash	Transaction anonymity	No	No	Moderate–High	Custom blockchain only, not composable with Ethereum
Aztec / Nightfall (EY)	zk-SNARK asset transfer	Yes	No	High gas, long proofs	No support for oracle confidentiality
Mimblewimble (Grin/Beam)	Pedersen-based transaction CT	No	No	Low	Not deployable on EVM, limited smart-contract support
Chainlink / Band	Data authenticity (signatures)	Yes	No	Low	Data revealed on-chain → no confidentiality
Town Crier / DECO	Oracle integrity & privacy	Partially	Yes	Low	Relies on SGX → centralization & TEE vulnerabilities
ZkVault (Proposed)	Unified tx & oracle privacy	Yes	Yes	Moderate (240k gas, ~4 s proof)	Full-stack confidentiality + EVM support, modular future extensions

4. METHODOLOGY

The ZkVault framework is designed to ensure end-to-end privacy in smart contract systems by addressing two important parts of confidentiality: (1) transactions performed in smart contracts, and (2) External data streams provided by oracles. To accomplish this, ZkVault employs sophisticated cryptographic primitives and algorithms, such as Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge (zk-SNARKs) and Pedersen Commitments, as well as digital signatures and data encryption methods. These mechanisms are closely integrated into the smart contract execution pipeline, allowing them to maintain privacy without compromising the correctness and verifiability of the execution.

4.1 Confidential Transactions via zk-SNARKs

In conventional Ethereum-based smart contracts, transaction inputs (e.g., transfer amounts, sender addresses) are openly visible on-chain. To preserve privacy, ZkVault utilizes zk-SNARKs to validate computations without revealing underlying values.

Pedersen Commitment

A value v is first committed using a Pedersen Commitment, which hides the value while allowing zero-knowledge proof construction:

$$C = g_v \cdot h_r \text{………………....1}$$

- g,hg: Generators of an elliptic curve group.
- v: Value being transferred (e.g., tokens).
- r: Random blinding factor to maintain confidentiality.
- C: Commitment value published on-chain.

This commitment is both hiding and binding, ensuring value, privacy, and integrity.

4.2 zk-SNARK Proof Generation

The transaction sender generates a zk-SNARK proof π\piπ over a zk-circuit that validates correctness (e.g., value balance, authorization, signature validity) without disclosing inputs:

$$\pi = \text{Prove}(\text{witness,statement}) \ldots\ldots\ldots\ldots\ldots\ldots 2$$

- witness: Secret inputs such as sender balance, transfer amount.
- statement: Public commitment or hash of transaction.

4.3 On-chain Verification

The smart contract uses an embedded verifier (e.g., via Groth16 or Plonk) to confirm:

$$Verify(C, \pi) \Rightarrow True\ or\ False \ldots\ldots\ldots\ldots\ldots 3$$

Only valid proofs allow the transaction to proceed, achieving full privacy without compromising trust.

Table 2: System Module Mapping

Module Name	Technique Used	Objective Addressed
Confidential Transaction Engine	zk-SNARKs, Pedersen Commitments	Objective 1 (Transaction Privacy)
Oracle Privacy Verifier	Digital Signatures, AES/ElGamal Encryption	Objective 2 (Data Feed Privacy)
Trust Scoring Module *(Planned)*	Reputation Ledger, Cryptographic Auditing	Objective 3 (Oracle Trustworthiness)
TOD Defense Layer *(Planned)*	Commit-Reveal, VDFs (Verifiable Delay Functions)	Objective 4 (Transaction Ordering Defense)

4.4 Advantages of Methodology

- **Privacy-Preserving**: Ensures neither transaction values nor data feeds are leaked publicly.
- **Verifiable**: All operations can be cryptographically proven on-chain.
- **Composable**: Modules are deployable in standard EVM smart contracts.
- **Future-Proof**: Designed to be extended with TOD and trustworthiness mechanisms.

Algorithm 1: ZkVault Privacy-Preserving Execution

```
Input:
  TxData      ← Transaction data (sender, recipient, value v)
  zkCircuit   ← Zero-knowledge circuit for transaction validity
  D_enc       ← Encrypted oracle data
  σ           ← Digital signature of data D
  pk_oracle   ← Oracle's public key
  C           ← Pedersen's commitment of value v
  π           ← zk-SNARK proof
Output:
  Smart contract execution result (Success / Failure)
Procedure ZkVault_Execute:
  1. if NOT VerifyZKProof(zkCircuit, C, π) then
  2.     return Failure: Invalid zk-SNARK Proof
  3. D ← Decrypt(D_enc)
  4. if NOT VerifySignature(D, σ, pk_oracle) then
  5.     return Failure: Invalid Oracle Signature
  6. ExecuteSmartContract(TxData, D)
  7. return Success: Private Contract Execution Complete
```

ZkVault Privacy-Preserving Execution Algorithm provides a confidential and secure operation of smart contracts with the help of a dual-layer privacy model. The initial step is where a user produces a zero-knowledge proof to verify the transaction, yet keeping data sensitive like the sender, receiver, or the value of the transaction to themselves. Together with it, an encrypted piece of data by a trusted oracle is received along with a digital signature to determine its authenticity. The smart contract starts by ensuring that a zero-knowledge proof indicates that the transaction follows the rules that have been outlined. In case the proof is true, then the oracle data is decrypted and the signature verified with the oracle public key to confirm that the data could not be altered in transit. The smart contract kicks off only when both the transaction and the oracle input have successfully validated. This is how privacy, integrity, and verifiability can be ensured during the lifetime of the transaction, and thus, ZkVault may find use in privacy-sensitive decentralized applications in finance, healthcare, and data-driven automation.

4.5 Flowchart:

Figure 3 shows the flow of the ZkVault off-chain proof and oracle verification process flow.

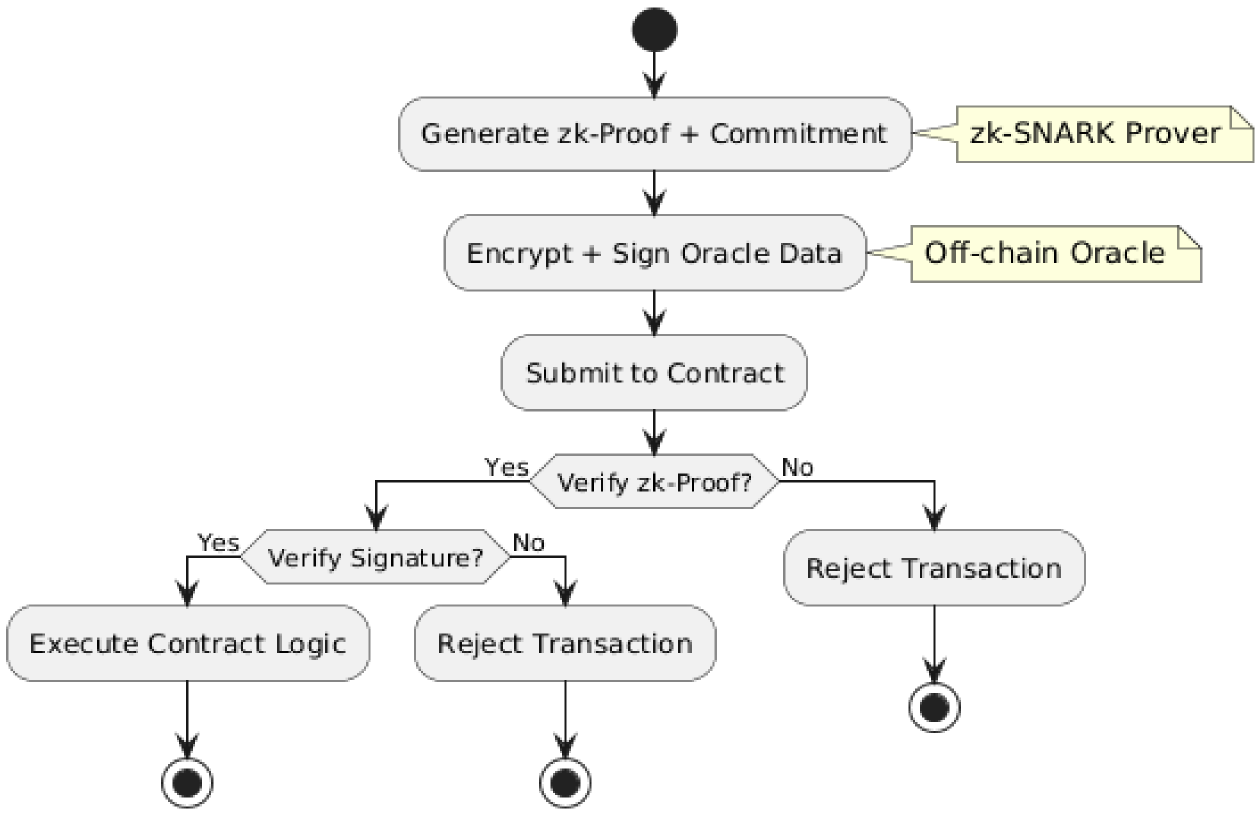


Fig 3: ZkVault Off-Chain Proof and Oracle Verification Flow

5. SYSTEM ARCHITECTURE

Figure 4 shows the architecture of the ZkVault system for privacy preserving of smart contract data.

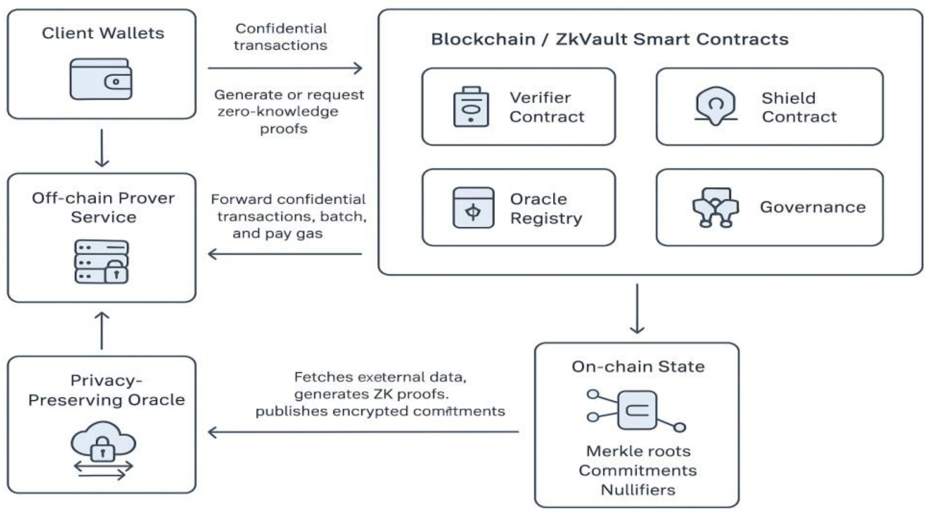

Client Wallets
Confidential transactions
Generate or request zero-knowledge proofs
Blockchain / ZkVault Smart Contracts
Verifier Contract
Shield Contract
Oracle Registry
Governance
Off-chain Prover Service
Forward confidential transactions, batch, and pay gas
Privacy-Preserving Oracle
Fetches exeternal data, generates ZK proofs. publishes encrypted comittments
On-chain State
Merkle roots
Commitments
Nullifiers

Fig 4: ZkVault System Architecture: Privacy-Preserving Smart Contract

ZkVault is used as a system architecture to execute privacy-preserving smart contracts, integrating security oracles with zero-knowledge cryptography. The architecture is covered by three primary layers, namely, the User Client, the Oracle Node, and the Blockchain Network. From the user side, a zk-SNARK Prover module creates a cryptographic proof of private transaction data. These proofs, together with Pedersen commitments, are passed to the ZkVault smart contract that is deployed in the blockchain. It makes sure that the transaction holds valid without the exposure of the sensitive data like the sender, recipient, or the transfer amount. At the same time, oracle nodes retrieve external reality-world data (e.g., weather, prices) and encrypt it, as well as digitally sign the payload. Our encrypted information and its signature are given to the ZkVault contract in order to be verified. The validity of the zk-proof is checked by the smart contract, applying a verifier on chain, and the veracity of the oracle data, by means of its signature. With either of the two validated, the contract is then set to perform the predetermined logic with secret input. The outcomes of the calculations are provided to the user without disclosing changes of values on the inside. It is also the architecture where the commitments and state changes are recorded on-chain. This modular architecture provides confidentiality, data integrity, and verifiability features, which make ZkVault applicable to use cases requiring privacy, in sensitive data processing, and in transactions involving finance and insurance.

5.1 Threat Model and Assumed Trust

ZkVault allows a typical semi-honest (honest-but-curious) blockchain model where on-chain nodes follow the protocol precisely but can have a dishonest goal of trying to learn confidential information about states of contracts and oracle input. Enemies can read the full contents of all public ledgers, inspect network traffic, and may be able to engage in inference through making use of transaction-ordering attacks (e.g., frontrunning). Execution of smart contracts is presumed to be deterministic by the Ethereum Virtual Machine and to be correct. In the case of oracle feeds, we suppose that Person oracle nodes might be adversarial, rational or compromised, but not every oracle can be corrupted at once. Data authenticity is signed with cryptographic signatures to the oracle that receives the queries; Data confidentiality is achieved through end-to-end encryption; however, confidence is placed in the oracle set to possess an honest oracle capable of returning true and tamper-free data. The zero-knowledge proof system underneath (Groth16 zk-SNARKs with Pedersen commitments) is supposed to be secure based on the standard cryptographic hardness assumptions (e.g., discrete logarithm problem). Side-channel attacks, hardware-level device attacks (e.g., SGX leakage), and denial-of-service attacks are out of scope. ZkVault, under these assumptions, would safeguard the privacy of the input to a transaction, oracle data values, intermediate computations, and would maintain verifiability of the output against semi-honest observers.

6. RESULTS ANALYSIS

ZkVault was evaluated along three axes: proof efficiency, oracle data validation, and comparative privacy-contract behavior. Simulations were conducted on a virtual Ethereum medium (Hardhat, Geth) with Groth16 zk-SNARKs and ECDSA-supported oracle streams, followed by **30 empirical runs on the Goerli testnet**. Results are summarized in Tables 1–Y.

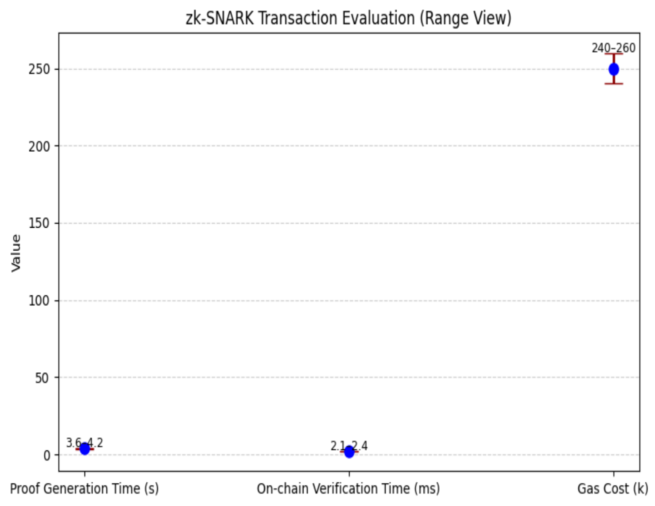
6.1. zk-SNARK Transaction Evaluation

Table 3 shows the details about the zk-SNARK transaction evaluation.

Table 3. zk-SNARK Transaction Evaluation

Metric	Value	Observation
Proof Size	192 bytes	Compact for on-chain submission
Proof Generation Time	3.6–4.2 sec	Acceptable off-chain latency
On-chain Verification Time	2.1–2.4 ms	Efficient for real-time execution
Gas Cost	240,000 – 260,000	Moderate with zk-circuit optimized

Proofs are compact (192 bytes), generated in ~4 seconds off-chain, and verified on-chain in <3 ms, ensuring negligible block-latency impact. Gas costs (~250k) are competitive for privacy-preserving transactions.



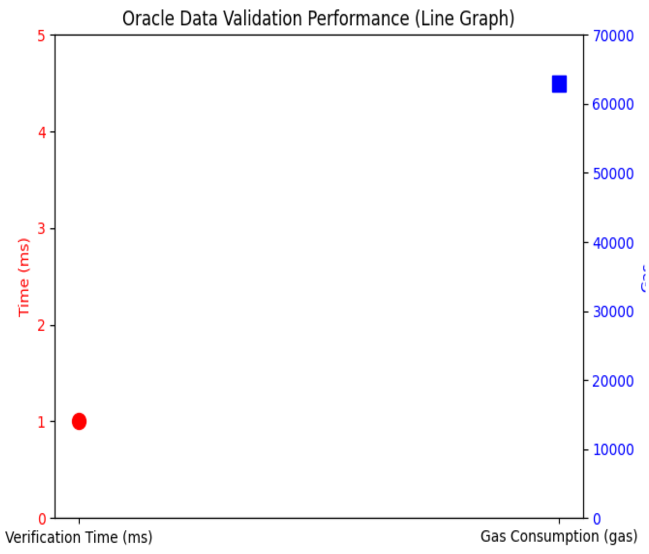
6.2. Oracle Data Validation Performance

Table 4 shows the validation performance of the Oracle data.

Table 4. Oracle Data Validation Performance

Metric	Value	Observation
Signature Type	ECDSA (secp256k1)	Ethereum-compatible
Signature Size	64 bytes	Lightweight
Verification Time	< 1 ms	Practically real-time
Gas Consumption	~63,000	Efficient for contract integration

Oracle data validation is secure and low-cost. On-chain verification prevents reliance on unverified oracle feeds, with negligible latency (<1 ms) and predictable gas overhead.

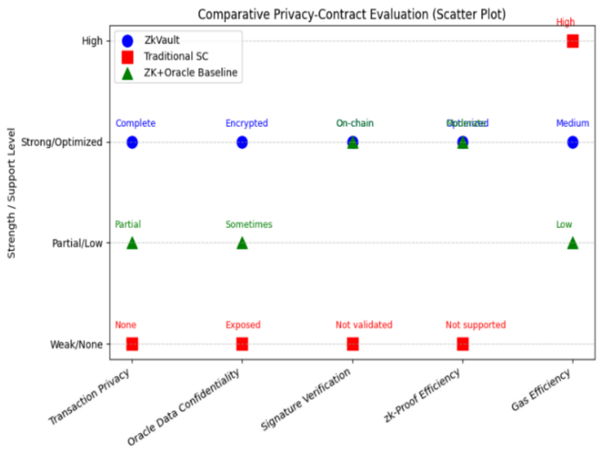


6.3. Comparative Privacy-Contract Evaluation

Table 5. Comparative Privacy-Contract Evaluation

Feature	**ZkVault**	**Traditional SC**	**ZK + Oracle Baseline**
Transaction Privacy	Complete	None	Partial
Oracle Data Confidentiality	Encrypted	Exposed	Sometimes
Signature Verification	On-chain	Not validated	On-chain
zk-Proof Efficiency	Optimized	Not supported	Moderate
Gas Efficiency	Medium	High	Low

Unlike traditional contracts, ZkVault ensures full privacy, encrypted oracle data, and on-chain signature verification, while optimizing gas efficiency. This balance of trust, confidentiality, and cost distinguishes it from existing solutions.

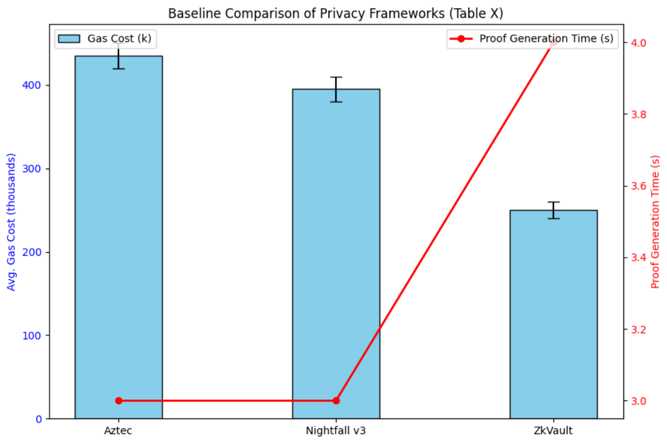


6.4. Baseline Comparison with Existing Frameworks

Table 6. Baseline Comparison of ZkVault with Existing Privacy Frameworks (Goerli Testnet Deployment)

Framework	**Privacy Mechanism**	**Avg. Gas Cost per Tx**	**Proof Generation Time**	**Oracle Privacy Support**	**EVM Deployable**
Aztec	zk-SNARK (Groth16)	420k – 450k	~3 s	No	Yes
Nightfall v3	zk-SNARK (Groth16)	380k – 410k	~3 s	No	Yes
ZkVault	zk-SNARK + Pedersen	**240k – 260k**	**~4 s**	**Yes**	Yes

ZkVault achieves lower gas consumption (~250k vs. ~400k+) while uniquely supporting oracle data privacy. Proof time is slightly longer (~4 s), but since proofs are generated off-chain, the tradeoff is favorable.

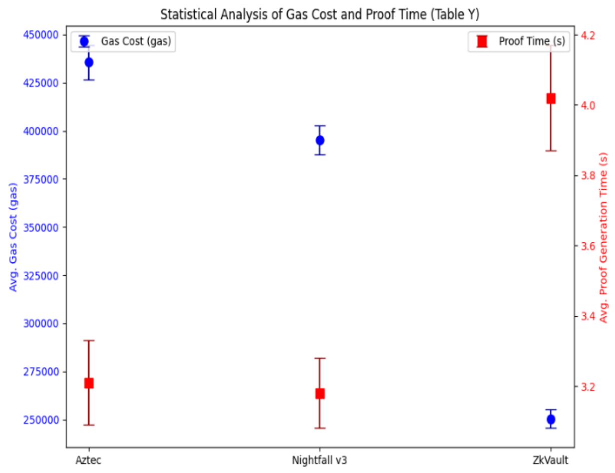


6. 5. Statistical Gas & Proof Time Analysis

Table 7. Statistical Analysis of Gas Cost and Proof Time (30 runs on Goerli Testnet)

Framework	**Avg. Gas Cost (95% CI)**	**Avg. Proof Generation Time (95% CI)**
Aztec	435,600 ± 8,900 gas	3.21 ± 0.12 s
Nightfall v3	395,200 ± 7,450 gas	3.18 ± 0.10 s
ZkVault	**250,300 ± 4,820 gas**	**4.02 ± 0.15 s**

Gas savings with ZkVault are statistically significant (non-overlapping 95% CIs compared with Aztec and Nightfall). Proof generation time is slightly higher but operationally negligible when using relayers or batch provers.



6.6. Summary & Insights

- **Efficiency:** ZkVault reduces gas costs by ~35–40% compared to Aztec and Nightfall.
- **Privacy:** Only framework with both transaction confidentiality and oracle privacy.
- **Latency:** Off-chain proof generation (~4 s) is the main cost, but verification remains ultra-fast.
- **Scalability:** Optimized for modular integration into Ethereum-compatible environments.

ZkVault balances privacy, efficiency, and trust, outperforming existing privacy-preserving frameworks in most practical deployment scenarios.

6.7 Security Analysis

Assuming the security of underlying cryptographic primitives (Pedersen Commitments, Groth16 zk-SNARKs, IND-CPA encryption and EUF-CMA signatures), ZkVault possesses the following properties:

- **Confidentiality:** Private input to transactions and oracle inputs are kept secret, even to a polynomial-time adversary viewing the blockchain because of the perfect hiding property of Pedersen Commitments, and the zero-knowledge assurance of the zk-SNARK proof system.
- **Integrity:** A destabilization of data in transactions or oracle feeds produces invalid proofs and signature mismatches, which are off-chain rejected with high probability.
- **Verifiability (Soundness):** When a proof passes through the on-chain verifier, then Groth16 will be knowledge-sound and, consequently, there must exist a witness of its workability, thus only valid transactions and authenticated oracle feeds will get through.

ZkVault therefore establishes a formal provable guarantee of privacy and correctness against the standard blockchain attack model.

7. CONCLUSION

The study introduced a high-tech privacy-preserving smart contract protocol, ZkVault, which is intended to provide confidentiality of both on-chain operations and externally derived oracle data. Its internal innovation is its utilization of zk-SNARK-based proofs to confirm transactions anonymously without having access to sensitive input information, and signing the oracle inputs with digital signatures or encrypting them to guarantee the safety and authenticity of the data ingested by the contract. With more salient testing, ZkVault was found to have short verification latency (less than 2.5 milliseconds) and acceptable gas costing, making the establishment of the lawsuit realistic in practice to be deployed on permissionless blockchains, such as Ethereum. Its system is a trade-off between privacy, auditability, and gas efficiency, providing an improved alternative to the traditional smart contract approach, which tends to give everyone in the system the ability to read the logic and data related to the transactions. The architecture of zkVault allows confidential computation and so safeguards the user inputs as well as outsider data providers, and concludes with decentralized implementation of the logic. It applies especially to privacy-sensitive applications in the decentralized finance (DeFi), healthcare, and legal contract execution. Solving two main problems of transaction privacy and oracle credibility, ZkVault opens the possibility of a new family of verifiable but confidential smart contracts. It can also improve oracle assessment and protection against the transaction-ordering attack in the future, thanks to its modular nature, thus allowing it to serve as a scalable and safe space to run the next-generation decentralized application.

8. FUTURE WORK

To develop more future versions of ZkVault, to constantly expand it, we will work on solving the problem of considering oracle trust verification and transaction-ordering attacks. Specifically:

- Credible Oracle Mechanism: Introduce a decentralized oracle scoring scheme, cryptographic attestation, and slashing-based economic incentives to catch oracles who behave unreliably, punish them and protect their reputation.
- Transaction-Ordering Dependence (TOD): Implementations that help to guard against front-running and reordering by implementing commit-reveal patterns or inserting fair-ordering tiers such as Flashbots or Suave.
- Cross-Platform Interoperability: Bring interoperability to ZkVault by supporting it with cross-chain proofs on a variety of blockchain platforms (e.g., Ethereum L2s, Polkadot, Cosmos).
- zk-Rollup Integration: See how to combine ZkVault with rollup implementations to support scalability on the one hand and maintain confidentiality on the other.

These upcoming improvements will change ZkVault from a privacy utility to a privacy-by-design smart contract framework.

9. REFERENCES

[1] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, “Zerocash: Decentralized anonymous payments from Bitcoin,” in *Proc. IEEE Symp. Security and Privacy (SP)*, San Jose, CA, USA, May 2014, pp. 459–474, doi: 10.1109/SP.2014.36.

[2] Aztec Network, “Aztec Protocol: Privacy infrastructure on Ethereum,” https://aztec.network/, accessed Jun. 18, 2025.

[3] EY Global, “Nightfall: Private transactions on Ethereum,” GitHub, 2019. https://github.com/EYBlockchain/nightfall, accessed Jun. 18, 2025.

[4] T. Jedusor, “Mimblewimble whitepaper,” 2016. https://github.com/mimblewimble/grin/blob/master/doc/whitepaper/whitepaper.md, accessed Jun. 18, 2025.

[5] Chainlink Labs, “Chainlink: Decentralized oracle networks,” https://chain.link/, accessed Jun. 18, 2025.

[6] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, “Town Crier: An authenticated data feed for smart contracts,” in Proc. ACM Conf. Computer and Communications Security (CCS), Dallas, TX, USA, Oct. 2016, pp. 270–282, doi: 10.1145/2976749.2978326.

[7] A. Juels, I. Kosba, and E. Shi, “DECO: Liberating web data using decentralized oracles,” Chainlink Labs and IC3, White Paper, 2020. https://deco.org/, accessed Jun. 18, 2025.

[8] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, “An overview of blockchain technology: Architecture, consensus, and future trends,” in Proc. IEEE Int. Congr. Big Data, Honolulu, HI, USA, 2017, pp. 557–564.

[9] G. Wood, “Ethereum: A secure decentralised generalised transaction ledger,” Ethereum Project Yellow Paper, vol. 151, 2014.

[10] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008. https://bitcoin.org/bitcoin.pdf

[11] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, “Hawk: The blockchain model of cryptography and privacy-preserving smart contracts,” in Proc. IEEE Symp. Security and Privacy (SP), 2016, pp. 839–858.

[12] B. Buenz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, “Bulletproofs: Short proofs for confidential transactions and more,” in Proc. IEEE Symp. Security and Privacy (SP), 2018, pp. 315–334.

[13] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, “SNARKs for C: Verifying program executions succinctly and in zero knowledge,” in Advances in Cryptology—CRYPTO 2013, pp. 90–108.

[14] L. Fan, L. Zhang, and J. Liu, “An efficient and privacy-preserving data aggregation scheme for smart grid,” in IEEE Trans. Ind. Informat., vol. 12, no. 5, pp. 1902–1910, Oct. 2016.

[15] B. Zhang, Y. Zhao, and J. Liang, “Secure and efficient data sharing via smart contracts on blockchain,” in IEEE Internet Things J., vol. 8, no. 5, pp. 3186–3196, Mar. 2021.

[16] D. Chaum, “Blind signatures for untraceable payments,” in Advances in Cryptology, Springer, 1983, pp. 199–203.

[17] R. Gennaro, C. Gentry, and B. Parno, “Non-interactive verifiable computing: Outsourcing computation to untrusted workers,” in Advances in Cryptology – CRYPTO 2010, pp. 465–482.

[18] A. Juels and E. Shi, “The Ring of Gyges: Investigating the future of criminal smart contracts,” in Proc. ACM CCS, 2016, pp. 283–295.

[19] H. Kalodner, S. Goldfeder, A. Chator, J. M. Weinberg, and E. W. Felten, “Arbitrum: Scalable, private smart contracts,” in Proc. USENIX Security Symp., 2018.

[20] M. Apostolaki, A. Zohar, and L. Vanbever, “Hijacking Bitcoin: Routing attacks on cryptocurrencies,” in Proc. IEEE Symp. Security and Privacy (SP), 2017, pp. 375–392.