perfect-privacy)

Privacy Essentials

Your start on privacy design for Secret smart contracts Secret Network provides all the tools necessary to build fully private decentralized applications, but contracts do not automatically inherit perfect privacy. Privacy has a cost. Developers have both the freedom and the burden of choosing a privacy model that fits their applications' needs.

Protecting sensitive user data is not even the whole story when it comes to Secret dApps. Achieving a proper level of privacy can be mission critical in certain areas, such as competitive gaming, where an information leak could lead to a compromised system, say, if an attacker were able to cheat.

As a developer, it is your responsibility to understand the gamut of privacy risks, their implications, and how to best mitigate them.

Perfect Privacy

Let's start off with a thought experiment. In an ideal world, a perfectly private application would be one that an attacker could not plausibly deduce any information about regarding its state, history, or participants. To an observer, any time a participant reads from or writes to the contract, it would be indistinguishable from every other action.

All outputs would appear to be random and unpredictable, with no discernible patterns that could be used to infer the underlying operations or data. On top of all that, the timing and frequency of executions would need to be consistent across the lifespan of the application.

Understanding The Risks

Most of the items below are covered in greater detail here: Potential Attacks. In reality, there are many forms of contextual data that an attacker can use to infer information, de-anonymize users, or compromise the privacy of applications purely through analysis:

- transaction sender's address, their public transaction history, the timing and frequency of their transactions, and their relation to other accounts on the network
- public inputs and outputs of an execution, such as sent or received funds (e.g., wrapping/unwrapping SCRT or IBC tokens)
- timing of an execution, its temporal locality to other transactions (e.g., clustering, dispersion, co-ocurrence), and an execution's order relative to other on-chain events
- size of the encrypted inputs and outputs of executions, shape and size of the event log, and public transaction metadata includinggas_used
- and nested calls to other contracts
- timing, frequency, shape and size of query requests and query responses
- storage access patterns when a contract reads from and writes to the encrypted application database

Additionally, there are active attacks that can expose unwanted leaks of information:

- executions that produce measurable side-effects
- tx replay attacks, side-chain attacks, the Millionaire's Problem
- targeted storage access pattern attacks

Mitigating The Risks

Its important to note that a few of the risks listed above can only be mitigated by user OPSEC. For example, the timing and frequency of executions, tx sender address, and relations to other accounts are risks typically assumed by end-users. In these cases, education is the best instrument to protect users' privacy.

Some of the other risks should be mitigated by client-side software, such as wallets and dApp front-ends. This includes padding inputs to executions and queries, being strategic about the timing and distribution of query requests, and creating workflows that allow users to more easily create identities that are not linked to their public account.

The remaining privacy risks are those which can only be mitigated through contract design. Developers should strive to make all queries and executions appear indistinguishable from one another to an outside observer. This includes:

- padding query and execution results (seepad_query_result

- andpad_handle_result
- from theSecret Toolkit
-)
- usingGas Evaporation
- to make all methods consume a consistent amount of gas
- adding decoy attributes to the event log when necessary
- understanding storage access pattern attacks and employing a system of decoys or oblivious data models (or even UTXO)

Find code implementation of the above privacy preserving methods under our Privacy design section. Previous TPS and scalability Next Access Control Last updated7 months ago