Senior Project Proposal

Yuhang Cui
Computer Science Advisor: Professor Fan Zhang
Math Reader: Professor Yair Minsky
Bioinformatics Advisor: Professor Hoon Cho

Introduction:

The goal of this project is to create a bug bounty program on the blockchain for genomics data analysis services to allow anonymous submission of data leakage proofs without revealing any private genome data.

Background and Motivation:

With the rising popularity of genomics research and publicly available genome data services, protecting the privacy of individuals contributing to genome datasets is of utmost importance. Many such services only provide summary-level data and use access-control mechanisms to prevent data leakage. However, such mechanisms could fail due to exploits and inference attacks. For example, genotype imputation services, which use statistical inference to fill in missing genotypes, are vulnerable to data reconstruction attacks with malicious queries [1].

Considering the severe privacy concerns of genome data leaks, meticulous measures should be taken to proactively identify and prevent such attacks. One way of discovering such exploits is through bug bounty programs, where attackers who manage to exfiltrate secret data can claim a bounty (i.e., monetary rewards) and thus are incentivized to alert the service provider, who can then fix the exploit and prevent further damage.

Implementing bug bounties in practice is often challenging, since the attackers may not trust the bounty manager to pay the bounty as promised. With the advent of blockchains and smart contracts, it is now possible to build bounties that avoid such trust.

Blockchain uses cryptography and randomized distributed algorithms to create a distributed append-only ledger, which is a growing sequence of data blocks with the hash value of the previous block, and the verified sequence has consensus from every participant. Many blockchains allow arbitrary programs, called smart contracts, to be executed on the blockchain. Hosting programs on blockchain have many advantages that are useful for fields like biomedical research, as analyzed in [2]. Some desirable properties of such systems that are especially relevant to this project are:

- **Availability**: Since blockchain is run by a distributed network of devices, it will always be accessible, in contrast to private servers that are prone to downtime due to failure and attacks.
- **Transparency**: Verified transactions are visible to everyone with access to the blockchain, preventing exploit submissions from being censored by the service provider.
- **Immutability**: Verified transactions on a block cannot be changed, which prevents any modification to previously verified data.

- **Secure Payment**: Many blockchains come with tokens or currencies that can be used for pseudo-anonymous payment, providing a secure payment mechanism to the exploit submitter.
- **Verifiability**: Many blockchain systems are designed to be verifiable on a consumer computer, providing a way for the submitter to verify the integrity of the smart contract.

These properties also pose some challenges, the most apparent one being the exploit submission being viewable by everyone on the blockchain, so the program needs a way to verify the exploits without exposing any private genome data. The distributed nature of blockchains also makes running smart contracts very expensive, as many miners need to run the contract and reach a consensus on the result, thus limiting the amount of computation that can be done in the smart contract.

Problem Description:

The bug bounty smart contract should satisfy the following requirements:

- **Functionality:** When a proof of exploit is submitted and verified, a reward is given to the submitter.
- **Privacy-Preserving:** Only the host (and possibly the submitter) can access the genomic information.
- **Fair Exchange:** The host gets proof of exploit if and only if the submitter gets the bounty payment.
- **Security:** The program is secure against brute-force attacks, as well as any other attacks circumventing its intended purpose.

Certain properties would be nice to achieve, although they are not strictly necessary:

- **Efficiency:** The smart contract should not be too computationally intensive, since running costs are expensive.
- **Verifiability:** A submitter can verify that the smart contract will work as expected. This may contradict certain security properties if encryption is required, but the program should be as transparent as possible to encourage participation.

Planned Approach:

Many blockchain primitives can help implement a bug-bounty smart contract. For example, the Sealed-Glass Proof paper provides a framework that uses hash locks to commit exploit submission, prevent front-running attacks, and enforce fair exchange [3]. However, such a protocol does not protect the submission data, so mechanisms to protect the private genome data need to be implemented.

Currently, the following technologies seem useful for solving the privacy-preserving problem:

- **Privacy-Preserving Smart Contracts:** Platforms like Oasis Network can limit access to a smart contract's internal state, allowing private data to be stored in the contract despite transactions being transparent [4].
- **Zero-Knowledge Proofs:** Protocols like zk-SNARK use a short and easily verifiable proof to check if the prover knows certain information [5]. This makes on-chain verification very fast, but generating the proof may be prohibitively slow.
- **Genome Encryption:** Some genome encryption protocols allow operations and tests to be performed on genetic data without revealing the information, which may be useful for the project [6]-[8].

The first few weeks will be spent researching relevant technologies and determining which ones are feasible for the project, then the solution will be implemented and analyzed for the deliverables.

Tentative Timeline:

- February 5: Finish initial literature and background research.
- February 12: Finish initial protocol designs.
- February 26: Partially implement initial design and finish midterm progress report.
- March 1: Midterm presentation to class.
- March 25: Finish program implementation.
- April 1: Finish debugging and documentation for code.
- April 8: Finish protocol analysis.
- April 19: Final report due for review. Finish math presentation preparation.
- April 22: Poster due.
- May 2: Final report website submission due.

Math Component:

The project report will include probability analysis on the security of algorithms and protocols used, investigations of novel mathematical aspects of these protocols, as well as an analysis of the final program stack as a mathematical system.

If time permits, the final presentation will also include a mathematical analysis of the blockchain system as the solution to the state machine replication problem under various distributed systems models.

Deliverables:

- Source code of the bug bounty program, and any associated code used for testing or analysis.
- Written report on design methodology and security analysis of the program.
- Slideshow and supporting materials for the presentation to the math department.
- Project poster for the poster session.

References

[1] Mosca, M.J., Cho, H. Reconstruction of private genomes through reference-based genotype imputation. *Genome Biol* **24**, 271 (2023). https://doi.org/10.1186/s13059-023-03105-6

[2] Tsung-Ting Kuo, Anh Pham, Maxim E Edelson, Jihoon Kim, Jason Chan, Yash Gupta, Lucila Ohno-Machado, The R2D2 Consortium , Blockchain-enabled immutable, distributed, and highly available clinical research activity logging system for federated COVID-19 data analysis from multiple institutions, Journal of the American Medical Informatics Association, Volume 30, Issue 6, June 2023, Pages 1167–1178, https://doi.org/10.1093/jamia/ocad049

[3] F. Tramèr, F. Zhang, H. Lin, J. -P. Hubaux, A. Juels and E. Shi, "Sealed-Glass Proofs: Using Transparent Enclaves to Prove and Sell Knowledge," 2017 IEEE European Symposium on Security and Privacy (EuroS&P), Paris, France, 2017, pp. 19-34, doi: 10.1109/EuroSP.2017.28.

[4] R. Cheng et al., "Ekiden: A Platform for Confidentiality-Preserving, Trustworthy, and Performant Smart Contracts," 2019 IEEE European Symposium on Security and Privacy (EuroS&P), Stockholm, Sweden, 2019, pp. 185-200, doi: 10.1109/EuroSP.2019.00023.

[5] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. 2012. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS '12). Association for Computing Machinery, New York, NY, USA, 326–349. https://doi.org/10.1145/2090236.2090263

[6] Emiliano De Cristofaro, Sky Faber, and Gene Tsudik. 2013. Secure genomic testing with size- and position-hiding private substring matching. In Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society (WPES '13). Association for Computing Machinery, New York, NY, USA, 107–118. https://doi.org/10.1145/2517840.2517849

[7] Cho, H., Wu, D. & Berger, B. Secure genome-wide association analysis using multiparty computation. Nat Biotechnol 36, 547–551 (2018). https://doi.org/10.1038/nbt.4108

[8] Froelicher, D., Troncoso-Pastoriza, J.R., Raisaro, J.L. et al. Truly privacy-preserving federated analytics for precision medicine with multiparty homomorphic encryption. Nat Commun 12, 5910 (2021). https://doi.org/10.1038/s41467-021-25972-y