DuckyZip: Provably Honest URL Shortening Using Smart Contracts and Verifiable Random Functions

Nadim Kobeissi[1]

Symbolic Software

Abstract URL shorteners are a common online service that allows the shortening of a long URL (often a Google Maps URL or similar) into a much shorter one, to use for example on social media or in QR codes. However, URL shorteners are free to behave dishonestly: they can, for instance, map a short URL into a long URL honestly for one party, while redirecting some other party into a different malicious long URL for the same short URL.
DuckyZip is the first provably honest URL shortening service which cannot selectively provide different "long URLs" to different parties undetected. DuckyZip uses a combination of Verifiable Random Function (VRF) constructions and a smart contract in order to provide a URL shortening service with strong security guarantees: despite the transparency of the smart contract log, observers cannot feasibly create a mapping of all short URLs to long URLs that is faster than classical enumeration.

Keywords: privacy-enhancing technologies, verifiable random functions, smart contracts

1 Introduction

As the internet landscape continues to evolve, so does the demand for efficient and secure web tools. One of these widely used tools is the URL shortening service, which turns lengthy URLs, like those from Google Maps[1], into concise links.[2] These services have been particularly useful for social media platforms, QR codes, and any other medium where space is a precious commodity.

However, the simplicity of URL shorteners belies a potential concern: the possibility of dishonest behaviors. Notably, these services could potentially present different URL destinations to different users from the same short URL. This raises important questions about the integrity and safety of these services as

[1] For example: https://www.google.com/maps/place/Ikebukuro+Station/@35.7295071,139.7060346,17z/data=!3m1!4b1!4m6!3m5!1s0x60188d5d4043e0dd:0x213775d25d2b034d!8m2!3d35.7295028!4d139.7109001!16zL20vMDIya3Zi?entry=tts, 198 characters.

[2] For example: https://t.ly/vjrx6, 18 characters.

they could be used to redirect unsuspecting users to malicious websites. This and other potential security issues that are caused by URL shorteners have been the subject of numerous studies. [6,7]

To address these concerns, this paper presents DuckyZip, a novel approach to URL shortening that ensures provable honesty and strong security measures. Unlike traditional URL shorteners, DuckyZip prevents selective delivery of different destinations for the same shortened URL and ensures that the mapping of short URLs to long URLs can't be deduced any faster than a classical enumeration process. The following sections of this paper will provide a comprehensive examination of DuckyZip's mechanics, its use of smart contracts and VRF, and how it establishes a new standard for honesty and security in the domain of URL shortening.

A working implementation of DuckyZip is available at `https://ducky.zip`, along with its source code.

2 DuckyZip Design

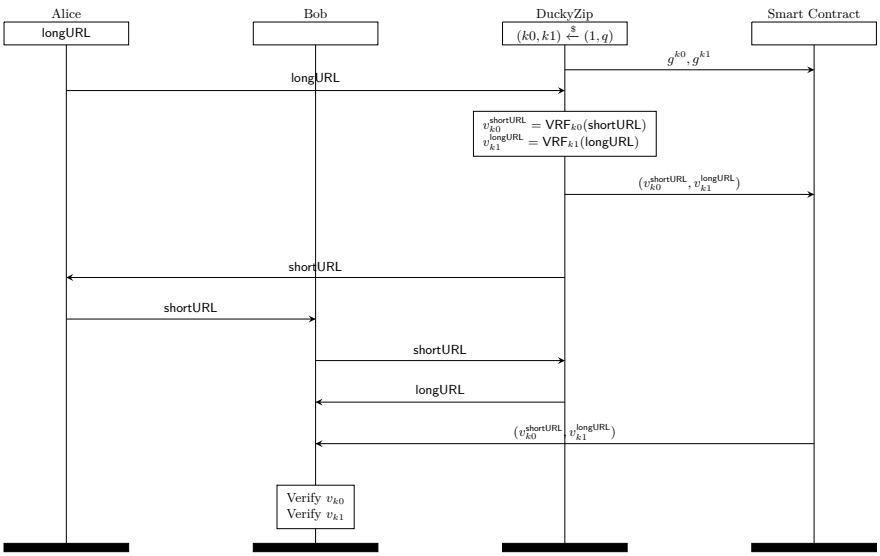

Figure 1. DuckyZip protocol description, where Alice is shortening a link and sending the short link to Bob so that he may "lengthen" it using DuckyZip.

DuckyZip depends on two cryptographic components: an authenticated append-only log and a Verifiable Random Function (VRF).

2.1 Designing the Distributed Append-Only Log

Certificate Transparency (CT) [4] attempts to enforce honesty with regards to SSL certificate issuance by having a number of certificate authorities and browser vendors maintain independent cryptographically authenticated append-only logs of all issued SSL certificates. This served as the initial inspiration for our design. However, since we do not possess the resources of Google or its certificate authority partners, we are unable to team up with a bunch of companies for our URL shortening project. We therefore use the Ethereum smart contract platform, which has recently become more practical due to the reduced gas costs.[3]

DuckyZip's smart contract contains the following data structures:

- **VRF public keys:** g^{k0} and g^{k1}.
- **VRF proof mapping:** a dictionary mapping the output from a VRF keyed with $k0$ to the output of a VRF keyed with $k1$.

2.2 The Need for a VRF

A natural question that may arise could relate to the need for a VRF in the first place. Why not just use the smart contract to map short URL strings to long URL strings?

While this would indeed accomplish DuckyZip's main goal of honest URL shortening and subsequent lengthening, the open nature of a smart contract's state would cause DuckyZip's entire history of short URL to long URL mappings to become public knowledge in real time. This could violate user privacy, since it allows anyone to see all the long URLs that are being shortened with minimal effort.

Our goal, therefore, is to make it so that retrieving a list of mappings from short URLs to long URLs is no less onerous in DuckyZip's case than it is in that of traditional URL shorteners: namely, the adversary would have to try all short URL strings one by one.

A naive approach towards accomplishing this could be to simply replace the mapping of $\mathsf{shortURL} \to \mathsf{longURL}$ with $\mathsf{H}(\mathsf{shortURL}) \to \mathsf{H}(\mathsf{longURL})$, where H is a secure hash function [2]. Realizing that such an approach is insufficient, one could propose to replace H with a maximally memory-hard password hashing function such as scrypt [8,1]. The VRF approach however provides stronger still security guarantees than the latter approach, without the associated performance impact.

We use the simple VRF construction proposed by Melara, Blankstein, Bonneau, Felten and Freedman [5], which we summarize here: for a group $\mathscr{G}$ with a generator g and of primer order q, the prover chooses a random $k \xleftarrow{\$} (1, q)$. The VRF also depends on two hash functions modeled as random oracles: one which maps to curve points ($\mathsf{H}_1 : \star \to G$), and one which maps to integers ($\mathsf{H}_2 : \star \to (1, q)$). The VRF is then defined as $v = \mathsf{VRF}_k(m) = \mathsf{H}_1(m)^k$.

To prove the correctness of the VRF output:

1. Prover chooses $r \xleftarrow{\$} (1, q)$ and transmits the values (v, s, t) where:
   - $v = \mathsf{VRF}_k(m) = \mathsf{H}_1(m)^k$
   - $s = \mathsf{H}_2(g, h, G, v, g^r, h^r)$
   - $t = r - s \cdot k \pmod{q}$
2. Verifier then checks that $s = \mathsf{H}_2(g, h, G, v, g^t \cdot G^s, \mathsf{H}_1(m)^t \cdot v^s)$

2.3 DuckyZip's Protocol

DuckyZip's operation can be described through the simple protocol shown in Figure 1:

1. Alice sends a long URL to DuckyZip for shortening.
2. DuckyZip shortens the URL, commits a set of VRF proofs to the smart contract, and sends the shortened URL back to Alice.
3. Alice sends the shortened URL to Bob.
4. Bob accesses the shortened URL and obtains the long URL from DuckyZip.
5. Bob is then free to query the smart contract (via Infura or similar if necessary) and to independently verify the existence and correctness of VRF proofs linked to DuckyZip's VRF keys and the short and long URLs relevant to this instantiation of the protocol.

Some practical considerations:

- DuckyZip's smart contract does not accept duplicate dictionary entries for the same short URL by not accepting the same VRF outputs. This works because same-key VRF outputs are deterministic.
- In practice, a short URL would be a 13-character string selected out of the set of allcase alphanumeric characters. This gives us a reasonably large search space for the short URL strings ($(26 \cdot 2 + 10)^{13} \approx 2^{80}$) while still allowing the full short URL to fit in a QR code without needing to increase the QR code's size or complexity.

The above protocol is reasonably performant and cost-efficient.

3 Conclusion

This paper introduced DuckyZip, a first-of-its-kind, provably honest URL shortening service, offering robust security guarantees. Leveraging Verifiable Random Functions (VRFs) and Ethereum's smart contract platform, DuckyZip is designed to prevent the selective provision of different destinations for the same shortened URL. Our design ensures that the mapping from short URLs to long URLs is confidential and authenticated, offering a significant improvement in user privacy.

Future work could involve further optimization of the DuckyZip system, and exploring its integration into existing web infrastructure. We also envisage extending the techniques used in this paper to other internet tools that may be vulnerable to similar types of attacks.

A working implementation of DuckyZip is available (or will be available) at `https://ducky.zip`, along with its source code.

Acknowledgements

We thank Lúcás Meier for his feedback throughout the authoring of this paper.

References

1. Alwen, J., Chen, B., Pietrzak, K., Reyzin, L., Tessaro, S.: Scrypt is maximally memory-hard. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 33–62. Springer (2017)
2. Aumasson, J.P., Neves, S., Wilcox-O'Hearn, Z., Winnerlein, C.: Blake2: simpler, smaller, fast as md5. In: Applied Cryptography and Network Security: 11th International Conference, ACNS 2013, Banff, AB, Canada, June 25-28, 2013. Proceedings 11. pp. 119–135. Springer (2013)
3. De Vries, A.: Cryptocurrencies on the road to sustainability: Ethereum paving the way for bitcoin. Patterns **4**(1) (2023)
4. Laurie, B.: Certificate transparency. Communications of the ACM **57**(10), 40–46 (2014)
5. Melara, M.S., Blankstein, A., Bonneau, J., Felten, E.W., Freedman, M.J.: {CONIKS}: Bringing key transparency to end users. In: 24th USENIX Security Symposium (USENIX Security 15). pp. 383–398 (2015)
6. Neumann, A., Barnickel, J., Meyer, U.: Security and privacy implications of url shortening services. In: Proceedings of the Workshop on Web 2.0 Security and Privacy (2010)
7. Nikiforakis, N., Maggi, F., Stringhini, G., Rafique, M.Z., Joosen, W., Kruegel, C., Piessens, F., Vigna, G., Zanero, S.: Stranger danger: exploring the ecosystem of ad-based url shortening services. In: Proceedings of the 23rd international conference on World wide web. pp. 51–62 (2014)
8. Percival, C., Josefsson, S.: The scrypt password-based key derivation function. Tech. rep. (2016)