# Limited use cryptographic tokens in securing cloud servers

Gautam Kumar, Brent Lagesse
Computing and Software Systems
University of Washington Bothell
{gautamk,lagesse}@uw.edu

## ABSTRACT

## INTRODUCTION

The essence of securing cloud systems is using multiple layers [9] of security to increase an attacker's cost for taking over the system. One of the possible layer of security is using moving target defences [3].

In this paper we propose an implementation of moving target defence using ephemeral servers and a central trusted authority which acts on behalf an ephemeral server and proxies requests to sensitive resources such as database servers, caching servers and REST end points. Hash chains are used as an authentication mechanism by the central trusted authority. We take advantage of the limited use property of hash chains to secure authenticate ephemeral servers for a limited period of time.

## BACKGROUND

### *Cryptographic hash function [11]*

A cryptographic hash function is any one way function which meets the following requirements

- Preimage resistance
- Collision resistance
- Second Preimage resistance

A hash function has preimage resistance if given a hash value $h$ it is computationally infeasible to find any message $m$ such that $h = hash(k, m)$ where $k$ is the hash key.

A hash function is collision resistant if, given two messages $m_1$ and $m_2$ it is hard to find a hash $h$ such that $h = hash(k, m_1) = hash(k, m_2)$ where $k$ is the hash key.

A hash function has second pre-image resistance if given a message $m_1$ it is computationally infeasible to find a different message $m_2$ such that $hash(k, m_1) = hash(k, m_2)$ where $k$ is the hash key. The second pre-image resistance is a much harder property to achieve for hash functions. This property is closely related to the birthday problem [7].

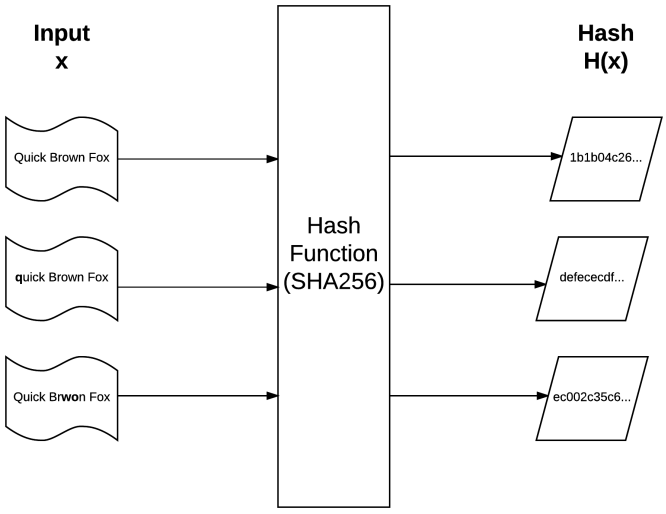


Fig. 1. A simplified view of a hash function which represents its input and potential result. The length of the hash sum always remains the same regardless of the input size. Any small change in the input drastically changes the output.

### *Hash Chains [4]*

Leslie Lamport [5] was first to propose the use of hash chains in his paper on a method for secure password authentication over an insecure medium.

"A hash chain is a sequence of values derived via consecutive applications of a cryptographic hash

function to an initial input. Due to the properties of the hash function, it is relatively easy to calculate successive values in the chain but given a particular value,it is infeasible to determine the previous value"

As an example, Let $x$ be the initial password a hash chain of length 2 would be $H^2(x) = H(H(x))$. So a hash chain of $n$ values is denoted as $H^n(x)$ and the $i^{th}$ value in the chain would be computed as $x_i = H(x_{i-1})$.

For a given value in the chain $x_i$ its computationally infeasible to determine the previous value in the chain $x_{i-1}$.

## Potential threats

According to OWASP's Top 10 security threats, "Sensitive data exposure" is the $6^{th}$ most critical type of security threat in web applications as of 2013 [14].

Sensitive data exposure simply refers to unintended exposure of sensitive information such as passwords, social security numbers, date of birth and so on. In the context of a cloud systems sensitive information may also include credentials to access a database, email server, REST API keys and so on. These credentials are usually stored as part of a configuration file which cloud servers can use to authenticate themselves with third party services within or outside the private cloud network.

According to a report by Risk Based Security [10] [12] the number of data leaks has dramatically increased from 2012 to 2013, to the tune of $812 million. Though sensitive data exposure in the context of cloud configurations would only constitute a small part of these leaks, leaking of such credentials can potentially lead to massive data leaks or other potential vulnerabilities being exposed to potential attackers.

Sensitive data exposure can potentially be a consequence of other threats such as cross site scripting (XSS) [8], Injection is the most critical threat while XSS is the $3^{rd}$ most critical threat as classified by OWASP in 2013 [14].

## Proposed solution architecture

The proposed architecture to defend against the threat of sensitive data exposure is to use a Central Trusted Authority (CTA) responsible for storing sensitive information. The CTA would act as a proxy and would make requests on behalf of client facing servers, refer Fig. 2.

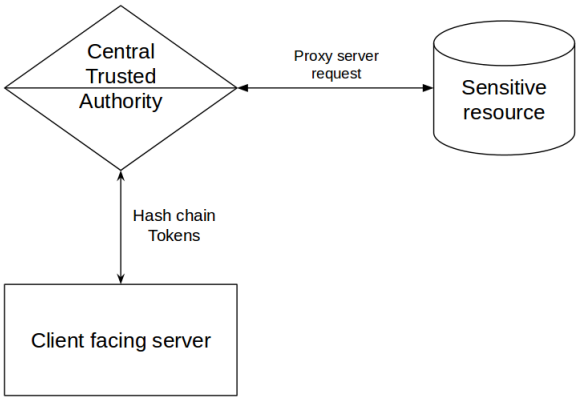


Fig. 2. Architectural overview of the system. This figure describes the three primary modules involved. The client facing server, The central trusted authority and a sensitive resource.

The client facing servers would use hash chains to authenticate with the CTA. Hash chains cryptographically limit the number of times a key can be used. Limited use was intentionally selected to promote the creation of a moving target for attackers.

### *Assumptions*

Client facing servers are the servers which are exposed outside the private cloud network environment. These client facing servers could potentially be load balancing servers, compute servers.

The client facing servers are assumed to be ephemeral. This is common in many cloud deployments [13] and contributes to the moving target nature of the security architecture. Companies such as netflix expect this behaviour with their chaos engineering architecture [1]. This allows for higher reliability of their server infrastructure.

Client facing servers are expected to shut down after their hash chain expires. This contributes to the ephemeral nature and also to moving target defence of the overall system.

### *CTA Architecture and Implementation*

The Central Trusted Authority consists of three primary components, A hash chain verifier, A storage backend and a request proxy. The CTA generally performs three roles within the system which are

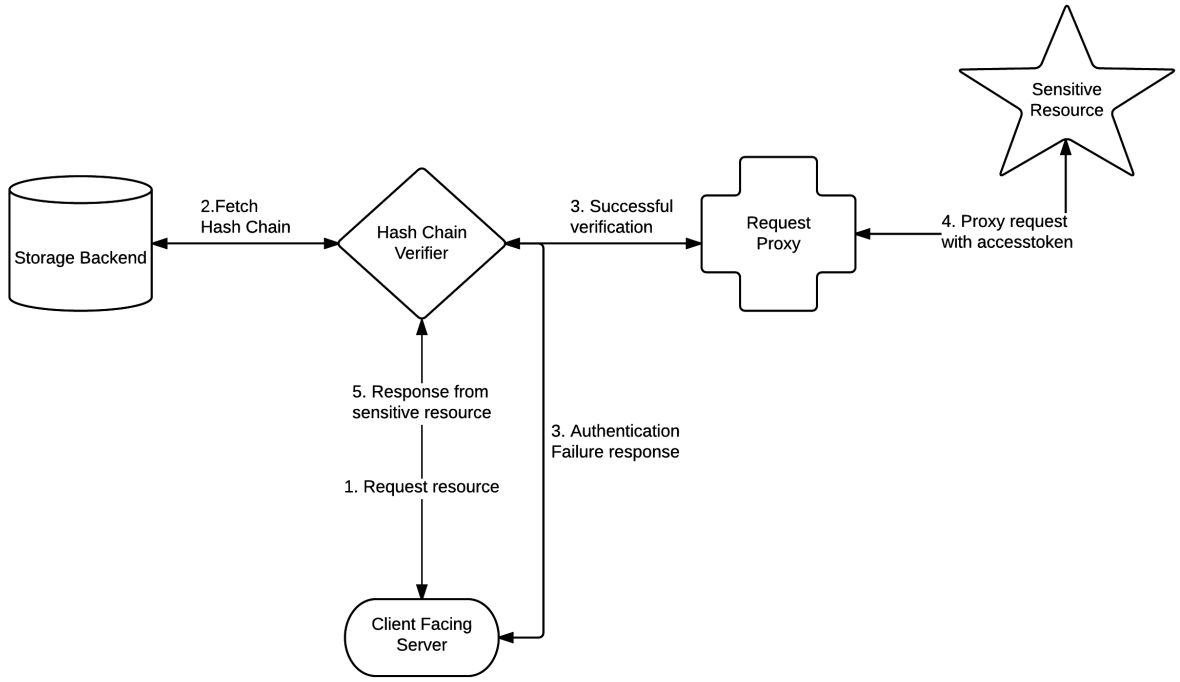


Fig. 3. Architecture of the Central Trusted Authority. The CTA consists of a storage backend, a hash chain verifier and a request proxy.

- Create new hash chain
- Verify hash chains
- Proxy requests for a client facing server.

*Creating new hash chain:* Hash chains are created by iteratively hashing a secret key $K$, $n$ number of times. After the hash chain is created the CTA stores $H^{100}(K)$ in the storage backend and returns $K$ and $n$ to the client facing server. The secret key $K$ is not stored by the CTA.

The client facing server can now use the the secret key $n$ number of times.

*Hash chain verification:* Hash chains are used to authenticate client facing server requests which require access to a sensitive resource. The hash chain verification process consists of four steps, Receive hash chain secret, compute the hash sum of the chain secret and verify with the storage backend.

The storage backend stores the last verified key $H^i$ of the hash chain where $i$ is the last verified index of the hash chain. When initializing the hash chain $i = n$.

- Receive hash chain secret $H^{i-1}(K)$ from the client.
- Compute $H^i$ by $H^i = H(H^{i-1}(K))$
- If the computed $H^i$ equals the value in the storage backend
  - Replace $H^i$ with $H^{i-1}(K)$.
- Else reject the key and refuse connection.

## PERFORMANCE TESTING

## POTENTIAL ISSUES

- Hash chain vulnerabilities [6]

## FUTURE WORK

- Timed release [2] -

## RELATED WORK

## REFERENCES


[1] A. Basiri, N. Behnam, R. de Rooij, L. Hochstein, L. Kosewski, J. Reynolds, and C. Rosenthal. Chaos engineering. 33(3):35–41.

[2] Konstantinos Chalkias and George Stephanides. Timed release cryptography from bilinear pairings using hash chains. In *Communications and Multimedia Security*, pages 130–140. Springer.

[3] David Evans, Anh Nguyen-Tuong, and John Knight. Effectiveness of moving target defenses. In Sushil Jajodia, Anup K. Ghosh, Vipin Swarup, Cliff Wang, and X. Sean Wang, editors, *Moving Target Defense*, number 54 in Advances in Information Security, pages 29–48. Springer New York. DOI: 10.1007/978-1-4614-0977-9_2.

[4] Dwight Horne. Hash chain. In Henk C. A. van Tilborg and Sushil Jajodia, editors, *Encyclopedia of Cryptography and Security*, pages 542–543. Springer US. DOI: 10.1007/978-1-4419-5906-5_780.

[5] Leslie Lamport. Password authentication with insecure communication. 24(11):770–772.

[6] D. Lee. Hash function vulnerability index and hash chain attacks. In *2007 3rd IEEE Workshop on Secure Network Protocols*, pages 1–6.

[7] Lawrence M. Lesser. Exploring the birthday problem with spreadsheets. 92(5):407–411.

[8] M. T. Louw and V. N. Venkatakrishnan. Blueprint: Robust prevention of cross-site scripting attacks for existing browsers. In *2009 30th IEEE Symposium on Security and Privacy*, pages 331–346.

[9] A. Panwar, R. Patidar, and V. Koshta. Layered security approach in cloud. In *3rd International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom 2011)*, pages 214–218.

[10] Risk Based and Security. An executives guide to 2013 data breach trends.

[11] Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In Bimal Roy and Willi Meier, editors, *Fast Software Encryption*, number 3017 in Lecture Notes in Computer Science, pages 371–388. Springer Berlin Heidelberg. DOI: 10.1007/978-3-540-25937-4_24.

[12] Xiaokui Shu, Danfeng Yao, and Elisa Bertino. Privacy-preserving detection of sensitive data exposure. 10(5):1092–1103.

[13] Luis M. Vaquero, Luis Rodero-Merino, and Rajkumar Buyya. Dynamically scaling applications in the cloud. 41(1):45–52.

[14] Dave Wichers. OWASP top-10 2013.