Efficacy of Schnorr Signature for Stateless Authentication

Prathyusha Chintala

Abstract

WebApplications/APIs accepts user credentials or api secrets and generate digital signature based state-less access tokens. The use of digital signature to create access tokens have the following advantages: 1) *Applications across data centers do not need to replicate the current user sessions.* 2) *Applications just need public key to verify the tokens as opposed to performing slow database query operations.* Most of the available frameworks (e.g.JWT) deploy traditional crypto-primitives such as RSA-DSS or ECDSA to create and verify the access tokens. Unfortunately, these algorithms are slow to create access tokens due to expensive modular exponentiations that involve the message. In this project, I evaluated the efficiency of Schnorr signatures for access token creation and verification. I compare the results against RSA-DSS and report that Schnorr signature improves performance by 166 X RSA.

1 Introduction

Web applications or APIs generally employ digital signature based access tokens to create user sessions. The user who holds the valid access token is allowed to access the services uninterrupted until the session expires.The following figure.1 depicts the general architecture deployed by the applications or API. The application is deployed across multiple data centers to achieve high availability of the service and the load balancer routes the requests to appropriate data center based on the load. For instance, user logged in one data center and further requests to may go to the second data center. How do we enable user to continue access to the service without forcing the user to login again?

Employing authentication service which issues digital signature based state-less token could help address the problem effectively. The below procedure(Shown in Figure.1) helps understand the effectiveness of state-less tokens.

1. User launches Web Application1 and enter username/passcode
2. Auth service receives the credentials and go to database and query to verify the credentials
3. Auth service contacts HSM to create digital signature based access token

4. Auth service responds to user with access token
5. User requests now sent to auth service hosted in data center2
6. Auth service verifies the access token using public key and contacts the API for the appropriate business functions

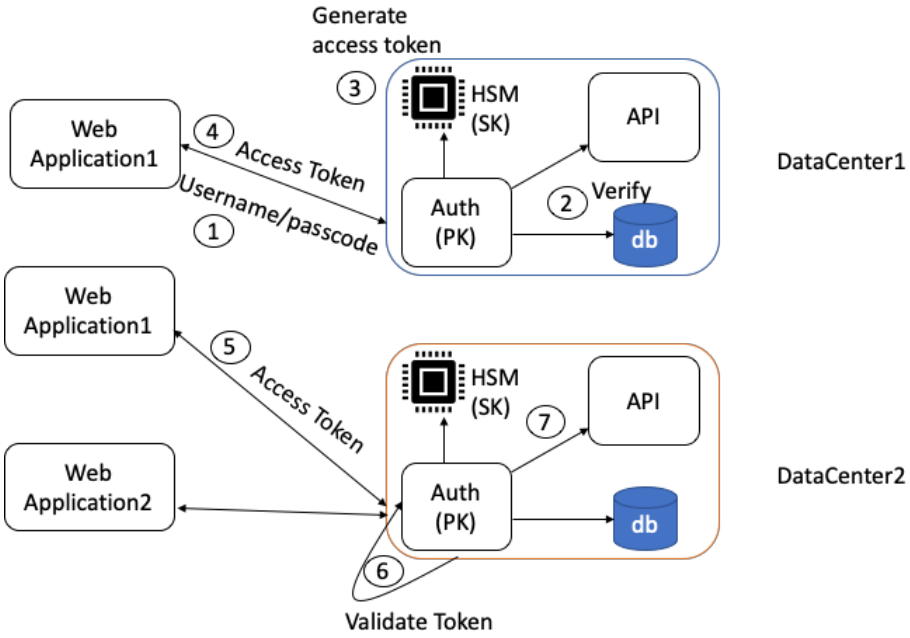

Figure 1: Web/API Authentication Architecture.

High throughput services(e.g. Facebook, LinkedIn etc.) require Auth service to be scalable and able to generate and verify millions of access tokens per second. The current widely known digital signature algorithms(e.g. RSA-DSS[8], ECDSA) are not capable of handling the load required by the posed services. The traditional algorithms can not scale because they require expensive modular exponentiations that involve message during the generation of access tokens.

Schnorr signature help address the scale problem effectively because it just require one multiplication and one addition to generate access token. In the next section, we explain the Schnorr signature and highlight it' s advantage over RSA.

2 Schnorr Signature Scheme

Schnorr Signature[1] scheme involves three steps, Key generation, Signature generation, Signature Verification.

- *Key generation:*

  1. Choose an elliptic curve E over a finite field $\mathbb{F}$q, where q is a prime number.
  2. Choose a random point P on the elliptic curve E.
  3. Choose a random integer 'a' from the range [1 to r], where r is the order of P. Calculate point Q = [a]P.
  4. Choose a hash function H.
  5. Public key p_k= (P, Q).
  6. Private key s_k=(a, p_k).

- *Signature generation*(s_k, m) :

  1. Choose a random integer 'k' from the range [1 to r], where r is the order of P. Calculate point R = [k]P.
  2. Calculate e = H($m \parallel R$).
  3. Calculate s = k+$a * e$ (mod r).
  4. Signature σ = (R, s).

- *Signature verification*(p_k, σ):

  1. Calculate e = H($m \parallel R$).
  2 If R + [e]Q = [s]P, then output "valid signature", else output "not a valid signature".

3 Schnorr advantage over RSA

Signature generation:

$$\begin{array}{lcl} Offline: & & Online: \\ \quad R = [k]P & & \quad \sigma = m^d (mod \quad n) \\ Online: & \Longleftrightarrow & \\ \quad s = k + a * e (mod \quad r) & & \end{array}$$

Schnorr **RSA**

- In the Schnorr signature generation, we could pre-calculate point R(offline calculation). By pre calculating point R, we are eliminating the time needed for point multiplication(R=[k]P) in the online calculation. We can create a linked list for pre calculated R values. For each signature generation we can get point R directly from the linked list. In online step we just need to perform one multiplication and one addition(s = k+$a * e$ (mod r)). But in the case of RSA we need to perform modular exponentiation, which takes more time than the Schnorr.

Signature verification:

$$R + [e]Q = [s]P \qquad \Longleftrightarrow \qquad m = \sigma^e (mod \quad n)$$

Schnorr **RSA**

- In the case of signature verification RSA is much faster than Schnorr. Because RSA need to perform modular exponentiation operation, where as in Schnorr we need two point multiplication and one addition operation(Which are much slower). In order increase the efficiency of Schnorr signature verification algorithm, we can use *batch validation technique*[7]. Here input to the signature verification function is public key, list of signatures(let say given 100). We can validate the given list of signatures all at a time due to the linearity of the Schnorr signature.

  Calculate e = H($m \parallel R$) for each given signature.

  E = $e_1 + e_2 ... e_{100}$(mod r). S = $s_1 + s_2 ... s_{100}$(mod r). Rs = $R_1 + R_2 ... R_{100}$(it is point addition).

  Finally calculate, if Rs + [E]Q = [S]P then output "valid signature", else output "not a valid signature"

4 Benchmarking

In Elliptic curve based Schnorr signature, I have chosen SPEC256K1 curve[4]. The recommended 256-bit, elliptic curve($y^2 = x^3 + ax + b$) domain parameters over a field $\mathbb{F}_{\text{ı}}$are,

p= FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F

a=0, b=7.

Base point g= 02 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798

Order of g is n= FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141

As shown in the figure 2, RSA- 4096 bit key can generate 50 tokens per sec and can verify 500 tokens per sec. where as, Schnorr can generate 8333 tokens per sec and can verify 12500 tokens per sec.

5 Conclusion

In this project, I have tested the performance of Schnorr signature for stateless authentication over RSA. Schnorr signature achieved the performance 166 X RSA. But there is a downside of using batch verification in Schnorr, If any signature in the batch is not valid then the whole batch is rejected. So the batch size is selected to be small.

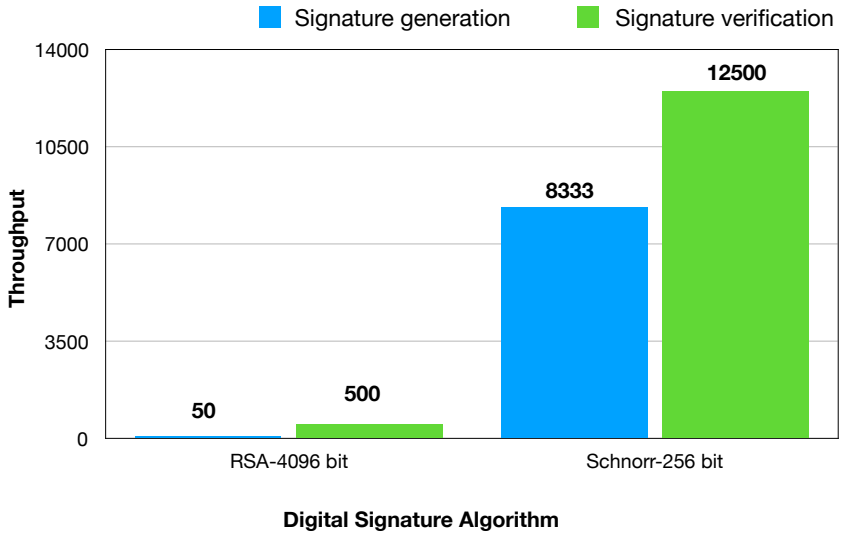

Figure 2: Benchmarking.

References

[1] Schnorr, Claus-Peter. *"Efficient signature generation by smart cards."* Journal of cryptology 4.3 (1991): 161–174.

[2] `http://andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-introduction/`

[3] `http://andrea.corbellini.name/2015/05/23/elliptic-curve-cryptography-finite-fields-and-discrete-logarithms/`

[4] `http://www.secg.org/sec2-v2.pdf`

[5] `https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03111/BSI-TR-03111_pdf.pdf?__blob=publicationFile#page=24`

[6] Martinez, V. Gayoso, and L. Hernandez Encinas. *Implementing ECC with Java Standard Edition 7.* International Journal of Computer Science and Artificial Intelligence 3.4 (2013): 134.

[7] Steven D. Galbraith. *Cryptography and Coding.* 11th IMA International Conference Cirencester 2007.

[8] Rivest, R.; Shamir, A.; Adleman, L. (February 1978). *"A Method for Obtaining Digital Signatures and Public-Key Crypto systems".*