Symmetric and asymmetric handshakes

André Brisson
Whitenoise Laboratories Canada Inc.
#701 – 1736 W. 10th Ave.
Vancouver, British Columbia Canada V6J 2A6
abrisson@wnlabs.com

Abstract—**Trusted computing and trusted cyber rely upon cryptosystems that are comprised of three parts: key creation, key management and key distribution.**

Prior to the advent of communications important keys like your birth certificate were provided manually. PKI asymmetric systems became the prevalent architecture around the time of radar because a mechanism was provided that allowed the distribution of keys in large communication (encryption or authentication) platforms. The flaws of that architecture were more of a nuisance at that point when compared to the benefit and lack of alternatives. Now, the ability to break and steal keys is an existential threat to that framework.

Distributed key systems (like one-time-pad enigma systems) languished because of the requirement for manual distribution of keys. That encumbrance for distributed, trusted cyber and trusted computing systems has been overcome. It is now simple, secure and online to create large, distributed authentication and encryption platforms that utilize one-time-pad distributed keys and where there is only partial disclosure of credentials.

This paper examines the comparison of asymmetric PKI and symmetric DDKI (Dynamic Distributed Key Infrastructure) handshakes. This paper also examines how to initiate secure communications with an endpoint/device/person that does not yet have a key without having to manually distribute the initial key.

Keywords—DIVA, DDKI, authentication, authorization, encryption, Whitenoise, identity, multi-factor, handshakes, trusted cyber, trusted computing, one time pad

I. INTRODUCTION

Distributed key virtual frameworks establish point-to-point encrypted tunnels. They can create and distribute session keys for dynamic and secure point-to-point connections for other network areas and uses such as key distribution. And, they can securely deliver an initial distributed session key online in order to enroll, authenticate and activate new endpoints/devices and persons with device specific keys.

In this architecture the server stores, creates, distributes and manages exponential "super" keys. Each key is unique to the endpoint. Each key is a unique deterministic random number generator that can create an unlimited number of one-time-pad keys (tokens). Each distributed key can in turn distribute more unique distributed keys.

The server can create dynamic session keys as well as securely distribute (one time) endpoint and server link keys for real-time enrollment of new persons, devices and networks. It also performs continuous, dynamic, one-time-pad authentication of all endpoints throughout a session.

One distributed key maintains continuous identity management and provenance of the endpoint and its data with one-time-pad authenticated encryption for data and one-time-pad, dynamic authentication for the endpoint, device or user with the DIVA protocol.

Because the key is a deterministic random number generator the same unique key can perform all cryptographic, key-based network security controls.

It is secure because it is a one-time-pad.

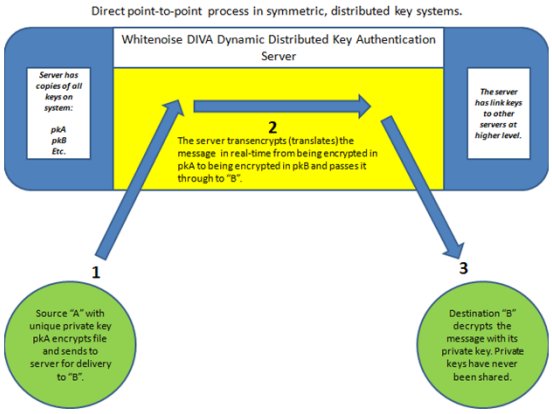
This advances trusted cyber and trusted computing because it prevents all known and anticipated cyber attack classes like man-in-the-middle and quantum computing attacks. It is easy to implement and understand, fast, and requires no training. It is interoperable and scalable for real-time enrollment of new endpoints and networks. Secure identities can be restored and refreshed online for potentially compromised identity i.e. Target, Home Depot etc. by simply resetting current dynamic offsets.

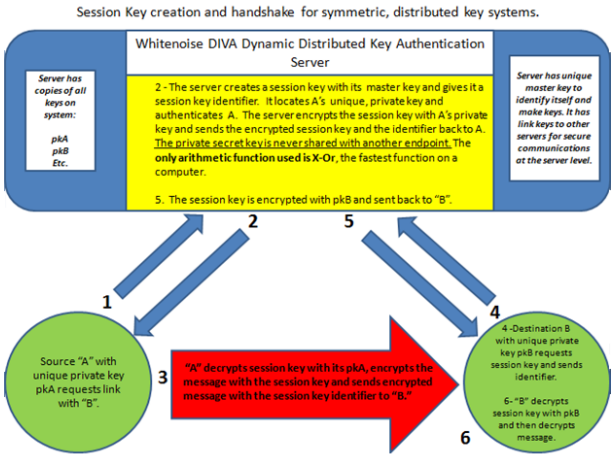
II. DYNAMIC DISTRIBUTED SYMMETRIC HANDSHAKES

In symmetric, dynamic, distributed key systems the server has copies of all the keys on a system. The keys are stored in an encrypted state. The keys are always kept separate from the last current dynamic offsets.

Each endpoint has only its unique, distributed, private/secret key. Secret keys are NEVER shared between endpoints. There is never key or offset exchange after setup.

The following illustration shows a system in its simplest configuration where endpoint A and endpoint B already have their own unique, secret keys and are part of a secure network.





1. The sender "A" wants to send a secure file to the receiver "B". The sender "A" encrypts a file with its own private key and sends it up to the server addressed to the receiver "B".

2. The server, which has copies of all the keys on this network, looks up and authenticates that both "A" and "B" are on the secure network. When the server has successfully authenticated both the sender "A" and the receiver "B" with DIVA the server transencrypts the file from being encrypted in the sender's key to being encrypted with the receiver's (B's) key.

3. The receiver "B" receives the file and decrypts it with its own private key. "A" and "B" have never shared keys or key materials.

III. DISTRIBUTED SESSION KEYS

It is desirable to be able to generate session keys in order to communicate with endpoints that do not yet have their own private/secret key. The purpose is to be able to establish secure communications with a new endpoint without first having to copy a key physically (manually) to that endpoint as has been traditional in distributed key systems. This allows simple, online scaling of the network.

It facilitates enrollment, authentication and secure, one-time key distribution. It facilitates the establishment of a secure point-to-point connection between endpoints without intercession (transencryption) by the server. The server continues to dynamically and continually authenticate the endpoints but no encrypted traffic is passing through it for potential capture.

"A" has a distributed key and wants to communicate securely with "B" who does not yet have a key.

1. "A" sends a request for a secure link with "B" who does not yet have a key.

2. The server generates a unique session key from its master key (unique to the server) and assigns a session identifier. The server locates "A"s unique private key and authenticates "A" with DIVA. The server encrypts the session key with "A's" private key and sends the encrypted session key and the identifier back to "A".

3. "A" decrypts the session key with its private key, encrypts the message with the session key, and sends the encrypted file and the session identifier to "B".

4. "B" sends the session identifier up to the server and requests the session key.

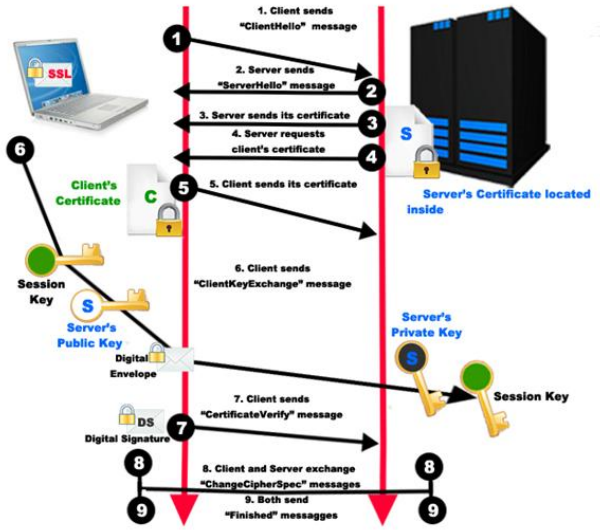
Note: At this stage for a new endpoint, online enrollment, authentication, and activation of the endpoint can be requisite to accomplish the one-time distribution of a private key and the endpoint DIVA application to "B". (There are different mechanisms to accomplish this i.e. Diffie-Helman, SSL etc.)

5. The server encrypts the session key with the private key of "B" and sends it back to "B".

6. "B" decrypts the session key with its private key, and then decrypts the file from "A" with the session key or establishes a persistent encrypted tunnel for direct, point-to-point, secure communications.

IV. PUBLIC KEY ASYMMETRIC HANDSHAKE

For comparison, let's look at the complexity of a typical asymmetric, public key process and key exchange handshake.



Its complexity is further aggravated by the very intensive mathematics required by PKI. It requires so much overhead in power, space and computational resources that it is literally unusable in many environments including the Internet of Everything where the majority of networked components operate under restrictive environments.

This is one mathematical technique for creating an asymmetric, public key session key. By comparison, after key load, the only operation used by DIVA and DDKI is X-Or. The either-or function is the fastest function available on a computing device.

Diffie Hellman Key Exchange

	Alice	Evil Eve	Bob
	Alice and Bob exchange a Prime (P) and a Generator (G) in clear text, such that P > G and G is Primitive Root of P G = 7, P = 11	Evil Eve sees G = 7, P = 11	Alice and Bob exchange a Prime (P) and a Generator (G) in clear text, such that P > G and G is Primitive Root of P G = 7, P = 11
Step 1	Alice generates a random number: X_A X_A=6 (Secret)		Bob generates a random number: X_B X_B=9 (Secret)
Step 2	$Y_A=G^{X_A}(\text{mod } P)$ $Y_A = 7^6 \text{ (mod 11)}$ $Y_A = 4$		$Y_B=G^{X_B}(\text{mod } P)$ $Y_B = 7^9 \text{ (mod 11)}$ $Y_B = 8$
Step 3	Alice receives $Y_B = 8$ in clear-text	Evil Eve sees $Y_A = 4$, $Y_B = 8$	Bob receives $Y_A = 4$ in clear-text
Step 4	Secret Key $=Y_B^{X_A}(\text{mod } P)$ Secret Key $= 8^6 \text{ (mod 11)}$ Secret Key = 3		Secret Key $=Y_A^{X_B}(\text{mod } P)$ Secret Key $= 4^9 \text{ (mod 11)}$ Secret Key = 3

The security of public key systems has always been flawed. PKI cannot prevent Man-in-the-Middle attacks and a host of other attack classes. In the past, computers were so slow that none of the attacks were considered feasible. Now computers are so fast it is simple to ravage our networks because of the lack of sustainable identity and data provenance and easily compromised security processes.

Dynamic Distributed Key Infrastructures (DDKI) and Dynamic Identity Verification and Authentication (DIVA) prevent all known and anticipated cyber attack classes.

V. GATEKEEPERS AND KEYVAULTS IN DISTRIBUTED SYSTEMS

An early deployment by A. Meyburgh at British Columbia Institute of Technology used an architecture comprised of Key Vaults and Gatekeepers.

"The GateKeeper and KeyVault servers can be used in any tier of network architectures traveling from IP to IP. It can be used from computer to computer, network to network, or computer to network. It can be used in any IP to IP context: wired-to-wired, wireless-to-wired, and wireless-to-wireless. These applications can work separately and independently but they obviously compliment each other."[1]

"The system is able to plug anywhere into a network easily because it relies on the data link layer between systems. Using the data link layer allows immediate integration with every IP based application with The KeyVault architecture to distribute session keys based on a distributed key. This enables point-to-point, dynamic connections that can be applied on other areas of the network apart from the tunnel."[2]

"No one GateKeeper can decrypt arbitrary data. When encrypted data needs to be decrypted, only the destination computer can decrypt it since only the two computers involved in the transmission can obtain the session keys from the KeyVault with no delay. The applications don't even know that the tunnel is there at all."[3]

"There are some implications in implementing a secure tunneling system (GateKeeper) combined with the KeyVault system. Not only does the system create a secure point-to-point communications layer but it also provides a way for dynamically adding new GateKeepers to the system without having to copy the key manually to every other client before communication can commence."[4]

VI. CONCLUSION

There is a need for rapidly scalable distributed platforms where there is only partial exposure of credentials. Distributed key systems previously required that a key be pre-authenticated and pre-distributed. The approach discussed enables the establishment of secure point-to-point communications with a party or endpoint that does not have a pre-distributed key.

The GateKeeper and KeyVault is a scalable symmetric key authenticated encryption system where no single trusted third party knows the final key.

This distributed key architecture advances trusted cyber and trusted computing. It can be used to create a hybrid system without direct integration with asymmetric public key infrastructures (PKI).

A dynamic distributed key extension eliminates system vulnerability to man-in-the-middle attacks. It has no exposed public key and mitigates against known attacks particularly quantum computing attacks because of the use of Whitenoise keys.

VII. ACKNOWLEDGEMENTS

Stephen Lawrence Boren, British Columbia, Canada

VIII. REFERENCES

[1] Albert Meyburgh, Distributed session keys - http://www.wnlabs.com/downloads/Tunnel_Distributed_Keys_distributing_more_keys.pdf

[2] André Brisson – Differences between PKI and DDKI handshakes - http://www.wnlabs.com/pdf/Comparison_of_handshakes.pdf

[3] http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf

[4] A.J. Brisson,"Cyber Belt presentation for NSA/NIST 2016 – Quantum computing attack resistant security" http://www.wnlabs.com/pdf/Cyber_Belt_Presentation.pdf

[5] André Brisson,"Key distribution method to shield service providers from key management - https://www.linkedin.com/pulse/key-distribution-paradigm-removes-management-from-carriers-brisson

[6] A.J. Brisson, "The Whitenoise Algorithm – A Visual Look," Technical Report, Whitenoise Laboratories, Vancouver, British Columbia, Canada, Nov. 2011. [Online]. Available: http://www.wnlabs.com/pdf/WhitenoiseAlgorithmVisualLook.pdf

[7] C. Coram-Mekkey, "Whitenoise Laboratories – An Overview," White Paper, Whitenoise Laboratories, Vancouver, British Columbia, Canada, June 2015. [Online]: http://www.wnlabs.com/papers/WhitenoiseOverview Short.pdf

[8] S.L. Boren and A.J. Brisson, "Dynamic Distributed Key System and Method for Identity Management, Authentication Servers, Data Security and Preventing Man-in-the-Middle Attacks," U.S. Patent Application 2009/0106551 A1, Apr. 2009. [patents granted]

[9] A.J. Brisson, "Dynamic Identity Verification and Authentication, Dynamic Distributed Key Infrastructures, Dynamic Distributed Key Systems and Method for Identity Management, Authentication Servers, Data Security and Preventing Man-in-the-Middle Attacks, Side Channel Attacks, Botnet Attacks, and Credit Card and Financial Transaction Fraud, Mitigating Biometric False Positives and False Negatives, and Controlling Life of Accessible Data in the Cloud," U.S. Patent Application 2013/0227286 A1, Aug. 2013. [patents granted]

[10] http://www.wnlabs.com/Presentations/Bringing_in_Legacy_Appliances_to_Secure_Networks.pps