P3KI

trust yourself.

Decentralized IPsec VPN using P3KI Core

Trust Scenario Whitepaper

1.6, 2018-04-12

Gregor Jehle (gregor@p3ki.com)

Contents

1 Introduction

1.1 Status Quo

Setting up and managing IPsec solutions currently heavily depends on centralized infrastructure to handle authentication and authorization of peers used in the establishment of Security Associations (SA). Protocols commonly used to facilitate this are IKE, KINK or IPSECKEY DNS records.

Centralized infrastructures are especially prone to attacks when considering adversarial scenarios such as (D)DoS – which is well within the reach of ambitious singular civilian entities – or targeted attacks by nation state actors. Disabling centralized infrastructure often has crippling results on the overall system.

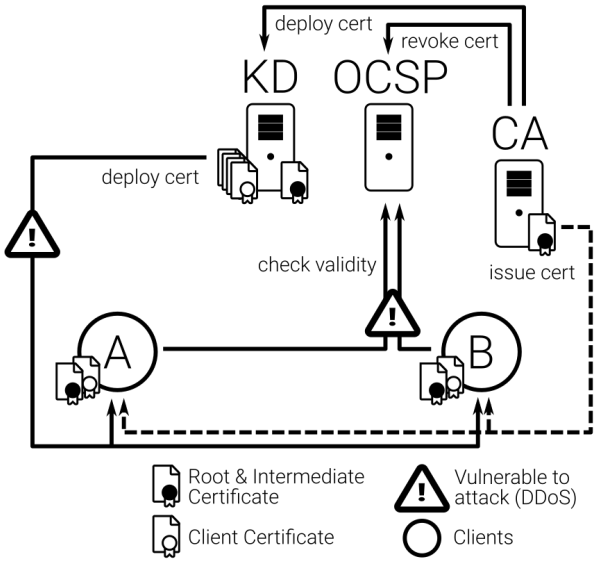

Figure 1: DDoS attack vectors in classic X.509 scenarios

1.2 Goals

This whitepaper outlines the possibility of designing a ***decentralized system for authentication and authorization***, supporting the ***delegation of trust in hierarchical and mesh networks*** while at the same time offering ***graceful degradation*** by employing concepts such as ***offline verifiable trust relationships***.

2 Solution

2.1 Focus

This whitepaper showcases possible improvements regarding key distribution and resiliency of IPsec applications. This is achieved by modifying common protocol suites responsible for establishing SAs to replace the part responsible for authenticating and authorizing peers with a P3KI Core-based solution.

2.2 P3KI Core Technology

P3KI Core is a storage independent Web-of-Trust solution offering the flexible delegation of trust, scenario-specific trust expressions of arbitrary granularity, and offline verification capabilities while easily integrating with existing solutions.

2.3 Relevant P3KI Core Functionality

2.3.1 Identities

Identities within the context of the ***P3KI Core*** technology are represented by the public key part of an asymmetric public/private key pair. The respective key pair must be compatible with a suitable signature scheme.

P3KI Core does not depend on a specific signature scheme. By design signature schemes are treated as replaceable modules which are easy to update. In fact, ***P3KI Core*** supports the usage of multiple signature schemes at the same time, which enables a smooth update path in case a certain signature scheme has been compromised.

By default, ***P3KI Core*** uses an ECC signature scheme based on curve 25519.

2.3.2 Expressing Trust

Trust within the ***P3KI Core*** Web-of-Trust is expressed using a concept called Trust Policy Languages (TPLs). TPLs are based on a mathematical lattice structure making them easily and provably verifiable.

TPLs are scenario-specific. This means that trust can be expressed as precisely as required without having to rely on predefined trust levels. TPLs can also be arbitrarily extended as application requirements change, without requiring updates to previously deployed clients. This enables the operator to introduce new trust levels at any time.

Trust is always expressed towards a specific identity (aka public key) and usually with validity limited to a specific time period. Any trust publicly expressed is signed by the identity issuing the respective trust information.

2.3.3 Trust Delegation

If Alice trusts Bob and Bob trusts Carol with a subset of the trust expressed by Alice, Bob effectively ***delegates Alice's trust to Carol***. The mathematical principles underlying the TPL design ensure that delegated trust is ***at most*** equal to what a given node is trusted with. On top of that, the number of acceptable hops over which a delegation may take place or whether a delegation is acceptable at all, is entirely under the control of the verifying party.

2.3.4 Authorization

The ***P3KI Core*** Web-of-Trust solution enables ***actual*** authorization scenarios.

Given that Alice trusts Bob and Bob similarly trusts Carol, Alice can verify the delegated trust chain to Carol without having to have prior knowledge of or direct trust towards Carol.

The only information required to perform this verification is the trust Alice locally holds towards Bob and the trust Bob expressed towards Carol. Bob's trust data also is not required to be queried from Bob directly. Trust data can be provided by any source, since all published trust data is cryptographically signed by its issuer (in this case Bob), the source can be totally untrusted since any modification of the trust data is impossible.

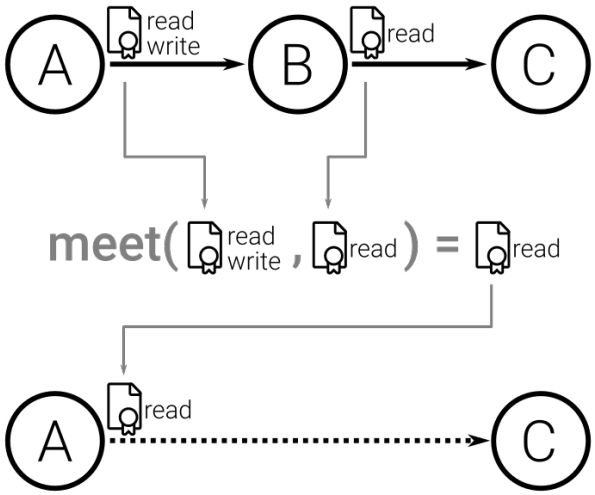

Figure 2: Authorization of ***C***arol by ***A***lice using delegation via ***B***ob

For a more practical discussion let's assume Alice trusts Bob with read and write access to her files. Bob in turn trusts Carol slightly less: Carol is only allowed to read Bob's files[1]. This effectively means that Carol should now have sufficient trust to also be able to read (but not write) Alice's files.

Once Carol wants to actually read Alice's files, she can quote Bob's trust and present it to Alice. Alice can combine Bob's trust – as quoted to her by Carol – with her local trust towards Bob and verify that sufficient trust exists to allow her files to be read by Carol.

This is possible even without Alice having to have prior knowledge of Carol.

2.3.5 Authentication

Authorization alone only establishes that verifiable trust of a specific level between any two parties via a number of hops exists. Since anyone can present (quote) trust expressed by anyone else on the network, this is not a sufficient guarantee to actually enable a given party to perform operations equivalent to the trusted level. Before this can happen, the trusted identity has to verify that it actually is the trusted identity.

In the case presented above, Carol now has to prove to Alice that she is in fact Carol. This is the case because in the previous ***authorization*** step we only established that a certain someone called Carol would have sufficient trust to execute the operation. By design, this is based on information anyone can present. Now the actual requester has to prove she's actually Carol (i.e. in control of Carol's identity).

To achieve this, a challenge/response protocol between Alice and Carol is performed in which Carol provides Alice with a cryptographic proof that Carol controls the private key for Carol's trusted public key.

It is important to note that these operations are possible without Alice previously or directly knowing Carol beforehand.

[1] In practice, the TPL would also allow Bob to restrict Carol's access to only a subset of files, so Bob could grant Carol access to his files without also delegating his access to Alice's files as well.

2.3.6 Updating Trust

Updating trust in the ***P3KI Core*** Web-of-Trust is simple and cheap. For instance, if Bob decides to trust Carol not just with read access to his files but also with write access, it's only a matter of adjusting the respective TPL expression, and publishing a new version of his updated trust data. This trust data needs to be suitably signed.

Now Carol can present this updated trust data, issued by Bob, to Alice and immediately gain read and write access to her files (remember, Alice already trusts Bob with read and write access to her files).

The significant difference to other existing solutions is that Bob – the de-facto analogue to an intermediate Certificate Authority (CA) in this scenario – can update his trust without affecting either Alice or Carol. It is not necessary to deploy a new certificate to Carol. Likewise, it also does not affect any trust Carol might have published herself.

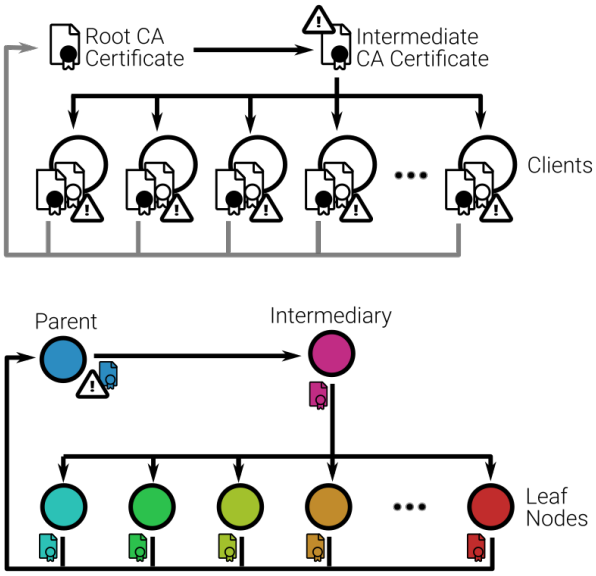

Figure 3: Parties involved in and affected by reacting to a compromised intermediary: X.509 (top), P3KI Core (bottom).

While Carol presents quoted trust to Alice, it is equally possible for Alice to query the trust network and fetch the trust data, issued by Bob, herself. At this point it can be illustrated again that the source of trust data is irrelevant.

This possibility is noteworthy when comparing ***P3KI Core*** to existing solutions such as ***X.509***. If an intermediate certificate authority (CA) in an X.509 hierarchy updates its trust to a given party, it is required to issue a new certificate. This becomes worse when an intermediate CA gets compromised and needs to be revoked; all trust issued by this intermediate CA becomes invalid. In conclusion all deployed certificates on leaf nodes must be updated, which can require a significant amount of time in large deployments[2].

In contrast, ***P3KI Core*** allows the roll-out of a replacement intermediate CA by publishing an updated version of the root CA's trust data. Leaf nodes are not affected at all in this scenario (there is no need to roll out new certificates to leafs). Instead, the next time a trust verification is executed, the updated trust data of root and intermediate CAs is used automatically. ***P3KI Core's*** Web-of-Trust architecture keeps the effects of such a compromised node local to the affected node.

2.3.7 Rescinding Trust

Updating trust data is trivial within ***P3KI-Core*** and works without interacting directly with other nodes. However, the concept of ***revoking*** trust within the ***P3KI Core*** Web-of-trust differs slightly from usual solutions as ***X.509***.

[2] Conservatively assuming that it takes each employee with VPN access only five minutes to install the updated certificate and 10,000 such employees, compromise of the VPN client intermediary certificate would require nearly five man months of deployment effort alone, to say nothing of the support costs as employees find that they can no longer connect to the VPN, which could easily triple that.

Short-lived Trust Since all published trust data should be tagged with a validity time range and updating trust is trivially easy and cheap, validity time ranges can be selected to be very short. In extreme cases, validity can be limited to seconds or minutes, instead of hours or days. Comparing this to X.509 deployments where a four-day validity range is already hailed as record-breakingly ***short-lived***[3], this is a major improvement, which offers complete new use cases.

Superseding Trust Trust issued by Alice is tagged with a counter value that increases every time a new version of her trust is published. If Bob is presented with two pieces of trust data issued by Alice, Bob is able to decide which one is more recent.

This allows nodes to track the latest known trust data and effectively react to the current trust data, presented to them by third parties. These third parties can present trust data that's currently valid but may have been already superseded by the issuer with a newer version.

For instance, Alice is now able to retrace that Carol presented a specific version of Bob's trust data to her and compares it to other instances of Bob's trust data she may know. She can even go as far as to ask the trust network for Bob's latest data and see if the network holds any trust data with a higher version counter value than the data presented to her by Carol.

Window of Compromise In adversarial conditions, checking the online trust network for the latest version of a given piece of trust data may not be feasible or possible. In such a scenario suitably short-lived trust should be used to limit the window of compromise.

[3] **Short-Lived Certificates @ Netflix** by Prabath Siriwardena https://medium.facilelogin.com/short-lived-certificates-netflix-fd5f3ae5bc9

2.3.8 Offline Capability

Since trust data is always verifiable to be correct – even in offline scenarios – and the way it is communicated or by whom, does not have any influence on its validity, ***P3KI Core*** allows modeling systems that work entirely offline. Extreme examples of this are sneaker-net and store & forward networks. Nodes participating in the Web-of-Trust can opportunistically exchange their latest view of what they think the trust network looks (even partially) with other nodes.

Trust data of a given identity is tagged with a monotonically growing counter value every time this identity decides to publish an updated version of its trust data. Afterwards, when presented with multiple versions of trust data, nodes in the trust network can decide, which piece of trust data of a given identity is the newest. This offers the following unique features:

- Trust data can be communicated between peers every time they have a chance to communicate
- Received trust data can be automatically merged with local trust data
- Resulting trust data collections represent a strictly equal or newer view of the trust network

Communicating trust data via central nodes or caching hubs is usually faster. P3KI Core trust data can be – similar to existing trust architectures – communicated this way. However, nodes can be easily designed to fall back to opportunistic communication with their immediate peers, if central services become unavailable.

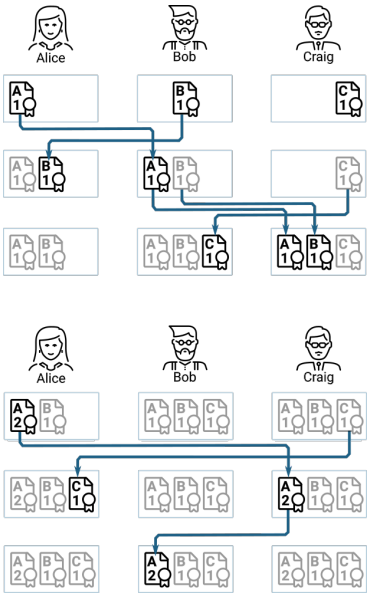

Figure 4: Opportunistic trust data exchange using P3KI Core.

Opportunistic Data Propagation Figure 4 demonstrates two example scenarios. In the upper half all parties generate their respective signed trust data, for sake of simplicity labeled as "generation 1". However, data is not published to any shared or central medium. Instead, it is transferred opportunistically every time two parties meet.

The first meeting is between Alice and Bob who simply exchange their personal data and learn about each other. Next up Bob and Craig meet. Bob can hand Alice's signed trust data to Craig without Alice having to be involved. At this point Bob and Craig have a full view of the "network's" state of trust, while Alice yet has to learn about Craig.

Opportunistic Update Propagation The lower half of figure 4 takes off where the upper half ends. Alice decides to update her trust data (e.g. add, remove, or update trust) and in doing so creates a new version of her trust data: "generation 2".

Alice now meets Craig and two things happen: For starters, Alice learns of Craig's generation 1 data. Craig already knows Alice because of previous communication he had with Bob. Craig can also now see that Alice has a newer version of her own trust data available, so Craig simply replaces Alice generation 1 data with her generation 2 data. In the last step shown, Bob and Craig meet yet again. This time Craig can hand Alice's generation 2 data to Bob and Bob is able to seamlessly update his trust data store with it without Alice having to be directly involved.

2.4 Implementation in the Context of IPsec

The ***P3KI Core*** features presented above will now be applied to the IPsec use-case.

Augmenting IKE with a ***P3KI Core*** Web-of-Trust backed distributed authorization and authentication network seems feasible now.

The ISAKMP parts of IKE, currently responsible for performing authentication based on X.509 certificates (e.g. RFC 2408[4] sections 1.5, 3.10, 3.11, 5.9, 5.10) as well as identification/authentication (e.g. section 3.8, 5.8), can directly be replaced with equivalent functionality backed by ***P3KI Core*** technology. The means of data communication, which need to be integrated, are scenario-specific and part of the application logic.

A direct implementation as part of an IKE-like userland daemon is one option. Alternatively, a compact stub interface that delegates execution of trust queries and verification to a ***P3KI Core*** daemon could be implemented. The functionality of both approaches is equivalent.

For instance, using strongSwan[5] this can easily be achieved by adding an EAP plugin interfacing with ***P3KI Core***.

Furthermore, implementing IKE with ***P3KI Core*** offers significantly improved flexibility and resilience when compared to the classic X.509 based implementation.

At the same time the trust model becomes exceedingly flexible allowing fully meshed scenarios unencumbered by strict hierarchical layouts, if so desired.

3 Conclusion

P3KI Core enables IPsec implementations to authenticate and authorize peers without having to rely on central infrastructures. Trust between peers can be modeled using expressions, which are specifically designed for the task at hand, thus not being limited by pre-defined trust levels or hierarchical structures as it is common with other solutions. In combination, the ***P3KI Core*** feature set enables a controlled and graceful degradation when operating in strong adversarial scenarios. It is also possible to verify trust data without requiring any online connectivity.

If this spiked your interest and you want to learn more about how our technology can solve your challenges, give us a call at +49 711 22 051 252, send us an email at contact@p3ki.com, or visit our website at p3ki.com.

[4] **RFC 2408** Internet Security Association and Key Management Protocol (ISAKMP) https://tools.ietf.org/html/rfc2408
[5] https://strongswan.org/

4 Appendix

4.1 Company Profile

P3KI GmbH is a subsidiary of the Berlin-based security consultancy ***Recurity Labs GmbH***. P3KI has been founded in 2014 focusing on creating ***P3KI Core*** the first truly flexible, distributed Web-of-Trust solution able to effectively handle real-world trust challenges in adversarial environments.

P3KI offers consulting services from analyzing existing or designing new concepts for trust systems up to customer-specific implementations and deployments of the ***P3KI Core*** technology targeting anything from embedded systems to large scale distributed Cloud setups.

You can reach us by phone at +49 711 22 051 252 (Stuttgart office) or +49 30 695 399 933 (Berlin office), via email by mailing contact@p3ki.com, and online at p3ki.com.

4.2 Glossary

Curve25519	**Curve25519** is an elliptic curve offering 128 bits of security and designed for use with the elliptic curve Diffie–Hellman (ECDH) key agreement scheme and the elliptic curve digital signature algorithm (ECDSA).
DDoS	A **Denial-of-Service attack (DoS attack)** is an attack where the perpetrator seeks to make a machine or network resource unavailable to its intended users by temporarily or indefinitely disrupting services of a host connected to the Internet. In a **Distributed Denial-of-Service attack (DDoS attack)**, the incoming traffic flooding the victim originates from many sources. This effectively makes it impossible to stop the attack simply by blocking a single source.
DNS	The **Domain Name System (DNS)** is a hierarchical decentralized naming system for computers, services, or other resources connected to the Internet or a private network.
ECC	**Elliptic-curve cryptography (ECC)** is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. ECC requires smaller keys compared to non-ECC cryptography (based on plain Galois fields) to provide equivalent security.
IKE	**Internet Key Exchange** (**IKE**, sometimes IKEv1 or IKEv2, depending on version) is the protocol used to set up a Security Association (SA) in the IPsec protocol suite.
IPsec	**Internet Protocol Security (IPsec)** is a network protocol suite that authenticates and encrypts the packets of data sent over a network.
ISAKMP	**ISAKMP** (**I**nternet **S**ecurity **A**ssociation and **K**ey **M**anagement **P**rotocol) is a protocol defined by RFC 2408 for establishing Security Associations (SA) and cryptographic keys in an Internet environment. ISAKMP only provides a framework for authentication and key exchange and is designed to be key exchange independent
KINK	**Kerberized Internet Negotiation of Keys (KINK)** is a protocol used to set up an IPsec Security Association (SA), similar to the Internet Key Exchange protocol (IKE), utilizing the Kerberos protocol to allow trusted third parties to handle the authentication of peers and management of security policies in a centralized fashion.

SA	A **Security Association (SA)** is the establishment of shared security attributes between two network entities to support secure communication. An SA may include attributes such as: cryptographic algorithm and mode; traffic encryption key; and parameters for the network data to be passed over the connection.
X.509	**X.509** is a standard that defines the format of public key certificates. X.509 certificates are used in many Internet protocols, including TLS/SSL, which is the basis for HTTPS, the secure protocol for browsing the web.