Z

Zero Trust Architecture

Dwight Horne
AT&T Center for Virtualization, Southern Methodist University, Dallas, TX, USA

Synonyms

Zero trust cybersecurity; Zero trust network; Zero trust threat model

Definition

A zero trust architecture is an abstract network design inspired by a zero trust threat model in which no inherent trust exists, and the network is always assumed to be hostile. In contrast to traditional, perimeter-based network security models that often statically define various trust zones and boundary choke points, a key goal of zero trust networks is that all users, devices, and network flows or transactions are rigorously authenticated and authorized.

Background

The traditional perimeter model in network security is somewhat comparable to the medieval perimeter model commonly used with castles. Tall castle walls protected the interior of the castle from the threats of the exterior, as the exterior was assumed hostile, while greater trust was afforded to the interior. In addition to the castle walls, there was often a moat, another layer of defense to further impede access. When coupled with these defenses, the castle drawbridge could be used to tightly authenticate and authorize traffic into and out of the castle. Similarly, corporate networks have commonly been divided into different security zones, each with a different implicit level of trust. In this perimeter model, network traffic at the boundary of each zone is tightly controlled with a device such as a firewall. Examples of zones commonly found in the traditional model might include the Internet or public zone, a demilitarized zone (DMZ), and one or more trusted Internal networks, which may also be further divided into additional trust zones or privileged zones. The assumption is that traffic or resources within a particular perimeter can be afforded a certain level of trust (or lack thereof), inherent by the fact that they exist in that zone.

Unfortunately, modern network threats, designs, and usage scenarios present a number of challenges to maintaining security with the perimeter security model. For example, mobile users and remote workers have become commonplace. Moreover, significant infrastructure has migrated from predominantly on-premises deployments to cloud-based solutions and software-defined networks. As the distinctions of traditional perimeters dissolve, defending a network with a perimeter-based security model becomes increasingly challenging. Another

© Springer Nature Switzerland AG 2025
S. Jajodia et al. (eds.), *Encyclopedia of Cryptography, Security and Privacy*,
https://doi.org/10.1007/978-3-030-71522-9

issue with traditional models becomes apparent when considering cases such as a compromised device due to exploitation of an application level vulnerability or a compromised user account from a successful spear-phishing campaign. When a network with perimeter-based security is compromised, lateral movement is simplified due to the implicit trust afforded to devices, users, and network traffic within the internal network. But designing networks instead with a zero trust mindset can accommodate modern paradigms with remote/mobile workers and multicloud infrastructure, while simultaneously mitigating threats such as a malicious device or compromised user credentials on a trusted internal network by in essence treating all hosts and services as though they are Internet facing.

Theory

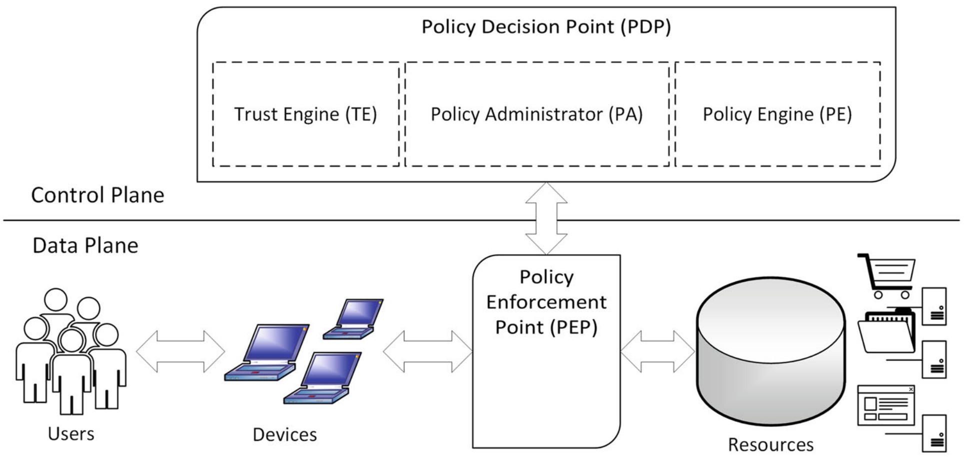
The core notion of zero trust (ZT) with use of that nomenclature goes back to at least 2002 when a study sponsored by the U.S. Army Medical Research and Material Command highlighted an intrusion containment approach targeting a ZT threat model where one assumes the system may have already been compromised and consequently cannot be trusted (Sood et al. 2002). In fact, the ZT threat model in that study postulated that all intrusion prevention methods will ultimately fail. The researchers proposed intrusion tolerance via hardware and software system components such as firewalls and servers that periodically restore themselves from a trusted source, which they referred to as self-cleansing systems. While the modern use of the term zero trust with reference to network architectures tends to convey a more extensive approach to network security, periodic restoration from trusted sources can be an element of device/software trust management in the context of ZT network architectures.

A more comprehensive interpretation of the ZT threat model applied to network architectures was popularized with the work of Kindervag (2010). Shifting from the old adage of "trust but verify" to a mentality of "never trust, always verify," ZT reimagines traditional hierarchical networks with perimeter-centric security models. Since that time, ZT theoretical frameworks have gained increasing attention across academia and industry. The ideas of ZT network architectures continue to evolve even after the appearance of more extensive knowledge bases such as the guide from Gilman and Barth (2017) as well as the National Institute for Standards and Technology (NIST) draft publication SP800-207 on zero trust architectures, the latter of which was approaching final form at the time of this writing (Rose et al. 2020). Unfortunately, since there are not yet widely recognized ZT architectural standards, information from different sources can vary in perspectives conveyed, terminology used, and subsets of zero trust principles that are emphasized. The remainder of this section seeks to harmonize the most prominent core principles, theoretical frameworks, and fundamental zero trust components from multiple sources to convey the essence of a current understanding of the state of the art in zero trust architectures.

Core Zero Trust Principles. There are a number of core principles that guide both the abstract conceptualization of a ZT architecture as well as implementations in varying contexts. Some of those core principles include:

- All users, devices, or other entities are strongly identified and validated (no implicit trust, Internet security everywhere).
- All network flows are strongly authenticated and authorized.
- A least privilege access control model is strictly enforced.
- Access to all resources is authenticated, authorized, and secured.
- Access is controlled dynamically, typically on a per session basis.
- Ensure that all devices are in the most secure state possible at all times.
- Inspect and log all activities.
- Ensure continuous evaluation rather than once-and-for-all trust.

Logical Components of a Zero Trust Architecture. A logical overview of fundamental components in a ZT network architecture is presented in Fig. 1. The first thing to notice is the clear distinc-

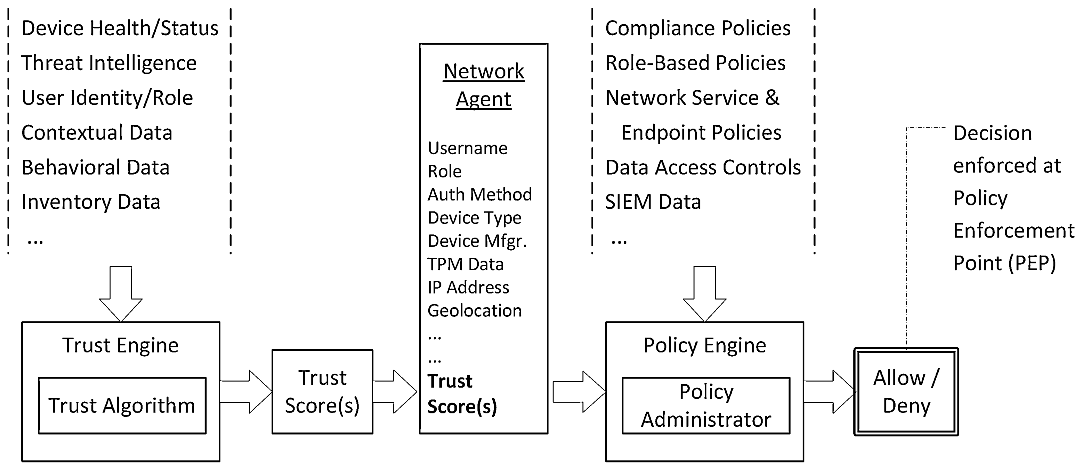

Policy Decision Point (PDP)
Trust Engine (TE)
Policy Administrator (PA)
Policy Engine (PE)
Control Plane
Data Plane
Users
Devices
Policy Enforcement Point (PEP)
Resources

Zero Trust Architecture, Fig. 1 Overview of zero trust architectural fundamentals

tion between the control plane and the data plane. The ZT components within the control plane must be protected as critical resources because they are used for trust management decisions to control what actions are allowed or disallowed in the data plane. In the control plane, an abstract concept often referred to as the Policy Decision Point (PDP) resides with logically separate functional subcomponents including the Trust Engine (TE), Policy Engine (PE), and Policy Administrator (PA). In reality, these logically distinct concepts could coexist within the same software set or be completely separate applications or services, and even geographically dispersed (although the latter may not be desirable). The TE accepts contextual inputs related to a particular network flow or resource access request and produces associated trust score data. The PE evaluates the breadth of data regarding a network access request with respect to access control-related policies and makes the final decision to allow or deny the request. The role of the PA is to execute the decision of the PE, ensuring that the decision is communicated to the Policy Enforcement Point (PEP) in the data plane. The PEP is responsible for allowing or disallowing the requested action, which may include dynamic reconfiguration of firewall rules, access of control lists, or other steps in concrete implementations. The PEP is also typically responsible for monitoring and terminating network connections and access to resources. Some have also promoted the notion of a separate monitoring plane. But a monitoring plane is not depicted in the diagram as consideration of the monitoring function embodied as a separate plane was not common practice at the time of writing.

Application

To gain a better understanding of how the theoretical zero trust (ZT) architectural components might map to actual implementations in real-world enterprises, consider the example presented in Fig. 2. In the example, the Trust Engine (TE) takes a variety of data inputs related to device health and status, user identity and role information, current threat intelligence, contextual information about the access request, behavioral data, and verified inventory data. The TE then applies its *trust algorithm* to compute trust score data. There could be one overall trust score, or independent trust scores, for instance, for user, device, and context. The resulting trust score(s) becomes part of the network agent request manifest. *Network agent* is ZT terminology adopted by Gilman and Barth (2017) to combine elements that would have traditionally been authorized separately such as the user, device, and application involved with the request. Subsequently, the network agent



Zero Trust Architecture, Fig. 2 Example of trust engine and policy engine roles in a zero trust architecture

request is considered by the Policy Engine (PE) and evaluated against various policies in effect that may include role-based access control policies, compliance policies (e.g., PCI, FISMA, HIPAA, ISO/IEC 27002, etc.), network service and endpoint policies, security information and event management (SIEM) system data, and more. The PE ultimately makes an authorization decision, which the Policy Administrator (PA) component conveys to the PEP for enforcement.

Very few enterprises may initially be able to fully realize a complete ZT network architecture in the near future. But many have already begun on their journey toward ZT, some even unknowingly, by beginning to institute ZT policies and controls. For example, many have already taken steps toward stronger multifactor authentication and/or are transitioning toward microperimeters, microsegmentation, and shrinking or elimination of implicit trust zones.

Google's BeyondCorp initiative was one of the first widely recognized corporate migrations toward a ZT network architecture within the context of existing enterprise infrastructure (Ward and Beyer 2014). The initiative moved away from a model that afforded a certain level of trust or privilege by virtue of being physically located on the internal corporate network, instead favoring an approach that ensures all access to enterprise resources is fully authenticated, authorized, and secured (encrypted). Some foundational elements of that migration that were described include securely identifying devices, securely identifying users, removing trust from the network, externalizing resources and workflows, trust inference for both users and devices, and implementation of strict access controls. A follow-on report from Osborn et al. (2016) provided further detail regarding trust calculations and access decisions that were based on the state of a device, user information, and the context of the request. After having completed a majority of the migration to ZT policies, the BeyondCorp planning and deployment steps described therein may benefit practitioners considering a similar migration to ZT architectures in other enterprise contexts.

Practitioners will find descriptions of multiple ZT deployment scenarios in sections 3–4 of the NIST SP800-207 draft. Section 7 also discusses migration plan considerations for greenfield projects, hybrid architectures, and ZT adoption in traditional networks with perimeter-centric security. The latter case may be the most commonly encountered scenario in enterprise networks at present. Consequently, the steps involved with a ZT network migration plan may often include:

- Inventory resources, data, and risks
- Identify all users, roles, and business processes/policies

- Perform risk assessment and policy revision as necessary
- Implement strong authentication of both users and devices
- Establish the trust evaluation approach(es)
- Inventory-desired traffic and network flows
- Involve users/consumers of data/services in the migration-planning process
- Design logical microsegmentation of applications, services, data, and resources along with a phased approach to achieving goals
- Implement per-session authorization of network flows and dynamic access controls
- Inspect, log, and analyze all traffic
- Continuously monitor, leverage analytics, adapt, and improve

ZT network architectures do not eliminate common security mechanisms such as firewalls, network monitoring, content filtering, access control lists, encryption, or intrusion prevention systems. However, in many cases they may be utilized in different ways. As ZT architectural conceptualizations are applied to varying networks, the implementation details and terminology used in specific circumstances may vary. For instance, the terminology used with presentation of BeyondCorp varied a bit from the more abstract concepts employed with the theoretical architecture of Fig. 1, but the essential elements and underlying principles were comparable. Different networks have disparate security requirements and other factors to take into consideration such as geographical dispersion, hybrid-cloud or multicloud infrastructure, technical debt from legacy applications and services, sensitivity of various enterprise resources, industry compliance requirements, and much more. But the fundamental tenets of the ZT threat model and ZT network architectures persist across diverse application domains. At its core, ZT is about avoiding implicit trust pools that provide gated access to resources based on network location. It is about dynamic access controls ensuring strict authentication, authorization, and security of all network flows given the realities of remote/mobile workers as well as cloud-based infrastructure and services. ZT is about ensuring confidentiality, integrity, and availability in a world where the network is always assumed to be hostile.

Open Problems and Future Directions

Standardization and Enhancements. The establishment of widely recognized open standards may help to facilitate adoption of ZT network architectures and the realization of associated benefits. Industry standards would also facilitate interoperability between tools and enablers from different vendors and provide a standard set of conceptualizations and terminology to foster related academic research and public discourse. Enhancements to the foundational ZT components will likely continue to be a focus going forward. For example, development of novel methods for dynamic access control policy definition and enforcement or improved trust algorithms would be beneficial. Similarly, security algorithms providing identity management, authentication, and authorization mechanisms that can ensure security against quantum-enabled adversaries will be of increasing importance going forward.

Attacks and Weaknesses. White hat security researchers as well as black hat hackers will undoubtedly concentrate on identification and exploitation of weaknesses in ZT networks. For instance, methods for subversion of ZT core components or processes will be a focus of study. ZT architectures are not a complete panacea and can be viewed as possibly increasing risk of single points of failure or higher value targets for exploitation. Moreover, if the implementation of ZT leads to greater technical complexity, it could conceivably result in increased risk. There may also be potential for increased risk of traffic analysis attacks with many ZT network implementations. For instance, consider observation of network flows to specific resources rather than opaquely multiplexed through a virtual private network (VPN)-tunneled connection. Yet another challenge may be with the requirement to inspect and log all data, and the resulting implications for potential storage requirements. The logs themselves may also become a valuable target for adversaries.

Tradeoffs and Varying Application Domains. Finally, most sources that address ZT architectures have done so in the context of traditional enterprise networks and resources. There are tradeoffs with ZT in the traditional context, including potential for increased complexity or challenges to usability (e.g., consider periodic user reauthentication requirements). But extending ZT technology to other domains such as low-power wireless sensor networks, Internet of Things (IoT) devices and 5G networks could introduce new tradeoffs and challenges or compound-existing problems. For this reason, future work may involve proposed changes for specific application domains and use cases or empirical comparisons evaluating associated tradeoffs. Future work may evaluate the extent to which hybrid ZT and perimeter-based security models might be more advantageous than pure ZT as well as the impacts of so-called megatrends such as machine learning (ML), artificial intelligence (AI), and big data on specific realizations of ZT network architectures.

Cross-References

- ▶ Authentication
- ▶ Authorization
- ▶ Continuous Authentication
- ▶ Entity Authentication
- ▶ Least Privilege
- ▶ Network Intrusion Detection System
- ▶ Public Key Cryptography
- ▶ Root of Trust
- ▶ User Authentication

References

Gilman E, Barth D (2017) Zero trust networks. O'Reilly Media, Inc., Sebastopol, USA

Kindervag J (2010) Build security into your network's DNA: the zero trust network architecture. Forrester Research, Cambridge, pp 1–26

Osborn B, McWilliams J, Beyer B, Saltonstall M (2016) BeyondCorp: design to deployment at Google; login, USENIX

Rose S, Borchert O, Mitchell S, Connelly S (2020) Zero trust architecture. Draft (2nd) NIST Special Publication 800-207, National Institute of Standards and Technology (NIST), U.S. Department of Commerce. https://doi.org/10.6028/NIST.SP.800-207-draft2

Sood AK, Huang Y, Simon R, White E, Cleary K (2002) Zero trust intrusion containment for telemedicine. George Mason University, Fairfax, USA

Ward R, Beyer B (2014) BeyondCorp: a new approach to enterprise security; login, USENIX

Zero Trust Cybersecurity

▶ Zero Trust Architecture

Zero Trust Network

▶ Zero Trust Architecture

Zero Trust Threat Model

▶ Zero Trust Architecture

Zeroization

Sean W. Smith
Department of Computer Science, Dartmouth College, Hanover, NH, USA

Definition

Zeroization refers to the process of automatically erasing or destroying data whenever a system transitions to a state in which an adversary might have access to this data.

Background

When computation involves data for which *confidentiality* is critical, security may require zeroization of this data.

Theory and Applications

Perhaps the most common scenario in which discussion of zeroization arises is high-security devices intended to resist physical attack by a dedicated adversary. For such devices, designers may consider various esoteric means (ranging from *crowbarring* power to ground on static RAM, to the use of explosives) to ensure that sensitive cryptographic secrets are destroyed.

However, the concept of zeroization is relevant even in more mundane computing scenarios. One aspect is *memory remanence*: the often surprising persistence of data storage in static and dynamic RAM across power cycling. (In the last decade, researchers demonstrated *cold-boot attacks* enabled by such remanence.) Long-term storage of data in static RAM can also result in *imprinting* that data, exacerbating the remanence problem. The emerging ubiquity of *FLASH* memory introduces new issues, since (due to the limited number of erase-write cycles in FLASH technology) FLASH memory systems typically implement a "rewrite" by simply writing the newer data to a different place and leaving the original data physically extant.

Many other standard features of modern computing systems also complicate zerioization. Does "deleting" a file actually result in its deletion from the medium underlying the file system? Does a virtual memory system reuse frames without first erasing the data? If a programmer carefully ensures a program overwrites sensitive data before transitioning to a less secure state, has the underlying compiler conspired to optimize away these operations by erroneously concluding that these writes are unnecessary, since they are never read?

Weingart (2000) and Chap. 16 in Smith and Marchesini (2008) provide good general discussions here (Gutmann 2001) discusses some issues with memory remanance.

Cross-References

▸ Confidentiality

Recommended Reading

Gutmann P (2001) Data remanence in semiconductor devices. In: Proceedings of the 10th USENIX security symposium. 2001

Smith SW, Marchesini J (2008) The craft of system security. Addison-Wesley, Prentice Hall

Weingart S (2000) Physical security devices for computer subsystems: a survey of attacks and defenses. Cryptographic Hardware and Embedded Systems – CHES 2000

Zero-Knowledge

Berry Schoenmakers
Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Eindhoven, The Netherlands

Editors note: *Due to the advancement of blockchain-based transactions, efficient noninteractive zero-knowledge methods, which are concise so that they fit inside a block, have been extensively investigated and deployed.*

Definition

Zero-knowledge is a property attributed to ▸ interactive proofs, ▸ interactive arguments, and ▸ noninteractive proofs. Just as the soundness property protects the interest of the verifier, the zero-knowledge property protects the interest of the prover. By means of a zero-knowledge proof, the prover is able to convince the verifier of the validity of a given statement, *without releasing any knowledge beyond the validity of the statement*. (Note that the notion of ▸ witness hiding proofs provides an alternative to the notion of zero-knowledge proofs.)

Theory

In other words, from executing a zero-knowledge ▶ protocol with an honest prover, the verifier should learn nothing beyond the validity of the statement. This is captured by stating that whatever the verifier "sees" when interacting with the prover by means of the zero-knowledge protocol can be *efficiently simulated* by the verifier itself. It is crucial to note that the zero-knowledge condition should be satisfied even if the verifier deviates from the protocol in arbitrary ways.

As a simple example, consider the following protocol. Let g denote a ▶ generator of a cyclic ▶ group G of order p, where p is a large ▶ prime number. A key pair is generated by choosing $x \in \mathbb{Z}_p$ uniformly at random, and setting $y = g^x$ as the public key and x as the corresponding private key (refer also ▶ modular arithmetic and ▶ public key cryptography). The protocol for proving knowledge of x on common input y runs as follows.

The prover chooses $u \in \mathbb{Z}_p$ uniformly at random, sets $a = g^u$, and sends a to the verifier. Next, the verifier chooses a challenge $c \in \{0, 1\}$ uniformly at random and sends c to the prover. The prover computes the response $r = u + cx \mod p$ and sends it to the verifier. Finally, the verifier checks that $g^r = ay^c$ holds.

The zero-knowledge property follows from the fact that the outputs of the following two probabilistic polynomial-time algorithms are identically distributed, where V^* denotes an arbitrarily cheating verifier. It is assumed that V^* is given as a rewindable black box. A triple (a, c, r) is called a *conversation*, as it consists of the messages exchanged during a run of the protocol.

Real Conversations

```
 Input: private key $x$
 Output: conversation $(a, c, r)$
1. Choose random $u \in \mathbb{Z}_p$
2. Set $a = g^u$
3. Send $a$ to $V^*$
4. Receive $c \in \{0, 1\}$ from $V^*$
5. Set $r = u + cx \mod p$
6. Output $(a, c, r)$
```

Simulated Conversations

```
 Input: public key $y$
 Output: conversation $(a, c, r)$
1. Choose random $c \in \{0, 1\}$, $r \in \mathbb{Z}_p$
2. Set $a = g^r h^{-c}$
3. Send $a$ to $V^*$
4. Receive $c' \in \{0, 1\}$ from $V^*$
5. If $c \neq c'$ rewind $V^*$ to point prior
to receiving $a$ and go to step 1
6. Output $(a, c, r)$
```

At step 5 of the simulation, the probability that $c = c'$ is exactly 1/2, since $c \in \{0, 1\}$ is chosen uniformly at random. Hence, on average two iterations are required to generate a simulated transcript (a, c, r).

The conclusion is that no matter what algorithm (or "strategy") a cheating verifier V^* follows in trying to extract useful information from the prover, the same algorithm can be used to generate identically distributed conversations without needing the cooperation of the prover. Whereas the real conversations are generated using the private key x as input, the simulated conversations are generated using only the public key y as input.

In general, the distributions of the real conversations and the simulated conversations do not need to be identical. *Perfect* zero-knowledge means that the distributions are indeed identical. *Almost-perfect* or *statistical* zero-knowledge means that the distributions are statistically indistinguishable (i.e., the statistical distance between the distributions is negligible). Similarly, *computational* zero-knowledge (refer also ▶ Computational Complexity) means that the distributions are computationally indistinguishable (i.e., cannot be efficiently distinguished).

By engaging in a zero-knowledge protocol multiple times, a cheating verifier may collect many valid conversations. In general, the simulation for a single run of a ▶ protocol can be easily extended to a simulation for multiple runs of the protocol as long as the runs are sequential, that is, the second run starts only after the first run is finished, and so on. In other words, the zero-knowledge property is preserved under *sequential composition*. However, *parallel composition*, where a prover is engaged in several runs of a protocol at the same time, in general, does not preserve the zero-knowledge property; running the above simulation k times in parallel does not result in an efficient simulation as the chances that $c = c'$ holds at step 5 for all runs at the same time will be only 2^{-k}.

The concept of zero-knowledge was introduced by Goldwasser et al. in the early 1980s (journal version appeared in (Goldwasser et al. 1989)). In (Goldreich et al. 1991), it was subsequently proved that a zero-knowledge interactive proof exists for every language in NP. *Noninteractive* zero-knowledge proofs were introduced in (Blum et al. 1988; Blum et al. 1991). There are many varieties of zero-knowledge proofs; see (Goldreich 2001) for an overview. Examples of some advanced notions are *concurrent* zero-knowledge (Damgård 2000; Dwork et al. 1998) and *resettable* zero-knowledge (Canetti et al. 2000).

Scalable noninteractive zero-knowledge proofs with particularly efficient verification were developed in (Gennaro et al. 2013; Parno et al. 2016). These *succinct* proofs, generally known as "zk-SNARKs," have been developed further in many works, enabling practical applications of zero-knowledge proofs for large, elaborate statements.

Cross-References

▶ Witness Hiding

Reference

Blum M, Feldman P, Micali S (1988) Non-interactive zero-knowledge and its applications. In Proceedings of the 20th ACM symposium on the theory of computing, pp 103–112

Blum M, De Santis A, Micali S, Persiano G (1991) Non-interactive zero-knowledge proof systems. SIAM J Comput 20(6):1084–1118

Canetti R, Goldreich O, Goldwasser S, Micali S (2000) Resettable zero-knowledge. In Proceedings of the 32nd ACM symposium on the theory of computing, pp 235–244

Damgård I (2000) Efficient concurrent zero-knowledge in the auxiliary string model. In: Preneel B (ed) Advances in cryptology – eurocrypt 2000, Lecture notes in computer science, vol 1807. Springer, Berlin, pp 418–430

Dwork C, Naor M, Sahai A (1998) Concurrent zero-knowledge. In Proceedings of the 30th ACM symposium on the theory of computing, pp 409–418

Gennaro R, Gentry C, Parno B, Raykova M (2013) Quadratic span programs and succinct NIZKs without PCPs. In: Advances in cryptology – EUROCRYPT 2013, Lecture notes in computer science, vol 7881. Springer, Berlin, pp 626–645

Goldreich O (2001) Foundations of cryptography – basic tools. Cambridge University Press, Cambridge

Goldreich O, Micali S, Wigderson A (1991) Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. J ACM 38(1):691–729. Preliminary version in 27th IEEE symposium on foundations of computer science, 1986

Goldwasser S, Micali S, Rackoff C (1989) The knowledge complexity of interactive proof systems. SIAM J Comput 18:186–208. Preliminary version in 17th ACM symposium on the theory of computing, 1982

Parno B, Howell J, Gentry C, Raykova M (2016) Pinocchio: nearly practical verifiable computation. Commun ACM 59(2):103–112

Zeta Function Computation

▶ Point Counting