ZKX™ TECHNICAL OVERVIEW

ZKX™ is a lightweight, platform-agnostic authentication solution suitable for use in tactical, strategic, and zero trust environments. ZKX is based on the mathematical principles of zero-knowledge-proofs, a methodical way in which one party, A (the Prover), can prove their knowledge or possession of some secret to another party, B (the Verifier), without revealing any bits of information about the secret.

Take, for example, the following question: Does numerical entry G belong within the set of numbers H? Traditionally, A could prove the state of G and H's relationship to B by revealing nothing except one bit of information – "yes", G *does* belong to H; or "no" G *does not* belong to H. However, formal advancements in the structure of zero-knowledge proofs have allowed A to prove their knowledge of G and H's relationship to B without revealing any information about it. ZKX utilizes these advanced cryptographic functions to achieve novel operational security in the context of multi-factor authentication (MFA).

One of the more popular methods of proving identity with zero-knowledge proofs is outlined by Feige, Fiat, and Shamir (FFS) (1987). In order to facilitate the FFS zero-knowledge proof (ZKP) scheme, a trusted center is required to publish a public Blum modulus, *N,* which itself is composed of two large prime numbers, *p* and *q,* both of the form 4r+3. *N* is public to all users in a given domain, and can be used universally, however no-one should know its factorization. Once this modulus is produced and published, this trusted computing center can be shut down, as it serves no further function in FFS.

Afterward this modulus is published, and users (provers) can register or enroll themselves in this FFS scheme by selecting *k* random numbers from the ring Z_N. These individual numbers are known colloquially as a user's "secrets" and are typically represented by S_k. All of a user's secrets are collected in one vector and are represented by $\mathbf{S} = \{S_1,\ldots, S_k\}$. In an analogy to the RSA cryptographic scheme, this vector would be the user's secret key. However, in terms of FFS and ZKPs of identity, it is more appropriate to think of **S** as a distinct collection of secret keys. After creating their vector of secrets, users can then calculate a vector of public data (effectively their public key), referred to as **V**. This public vector is calculated as **V** = **S**^{2} (mod N) = $\{S_1^2,\ldots, S_k^2\}$ (mod N). This public vector is then published; it will be used by the Verifier to validate the Prover's knowledge and possession of their secrets. Even though the public **V** was constructed from the private **S**, the secret values used within **S** are still secure, given that the numerical factorization of modular squares is a computationally hard problem. This stage of the FFS process is known as *enrollment*.

After a Prover is enrolled with the FFS scheme, they are capable of proving that their registered secrets are indeed in their custody to another party, the Verifier. After reading or receiving the Prover's public data, **V**, the Verifier can engage the Prover in ZKPs. First, the Prover selects a random number, r (a private value known as the *commitment* or *round ID*), which is then used to compute a new value, x (a public value known as the *witness*) in the form $x = \pm r^2$ (mod N). This witness is sent from the Prover to the Verifier. The Verifier then generates a k-length Boolean vector, $\mathbf{b} \in \{0,1\}^k$, and sends it to the Prover. This vector is referred to as the challenge. The role of the challenge is to select the public data which the Prover's identity will be challenged against. Likewise, the challenge informs the Prover which of their credentials they must supply in order to accurately respond to the Verifier's challenge. The Prover answers the Verifier's challenge by calculating its response, **y**, as: $\mathbf{y} = rS_1^{b1}S_2^{b2}\ldots S_k^{bk}$ (mod N). This response is returned to the Verifier, who can then check its validity with the following comparison: $\mathbf{y}^2 = \pm xV_1^{b1}V_2^{b2}\ldots V_k^{bk}$. If this equivalency is indeed true, this proves the Prover's knowledge of S as follows:

$$\mathbf{y} = rS_1^{b1}S_2^{b2}\ldots S_k^{bk} \pmod{N}$$
$$\mathbf{y^2} = (rS_1^{b1}S_2^{b2}\ldots S_k^{bk})^2 \pmod{N}$$
$$\mathbf{y^2} = (r^2)(S_1^{b1})^2(S_2^{b2})^2\ldots(S_k^{bk})^2 \pmod{N}$$
$$\mathbf{y^2} = x(S_1^{b1})^2(S_2^{b2})^2\ldots(S_k^{bk})^2 \pmod{N} \rightarrow \text{substitute } x = \pm r^2 \pmod{N}$$
$$\mathbf{y^2} = xV_1^{b1}V_2b^2\ldots V_k^{bk} \pmod{N} \rightarrow \text{substitute } \mathbf{S}^2 = \mathbf{V}$$

This process constitutes one *round* of FFS-based ZKP. In FFS, the probability that a malicious is user is able to successfully verify illegitimate knowledge is proven to be 2^{-kt}, where k is the length of a Prover's secret vector (**S**) and *t* is the number of ZKP rounds undertaken with the Verifier. In order to achieve efficient practical security, the configuration $kt = 20$ is put forward as a recommended standard.

During the enrollment phase of the FFS identity proofing scheme, users could populate their secret key vector with the outputs of other approved authenticators, credentials which they possess, or values derived from their credentials instead of randomly selected numbers. This would then yield corresponding public key values, represented in **V**, which are now exclusively tied to the user's identity. These values would then be published either publicly or specifically to some repository accessible to the Verifier. This data would then be used to authenticate the Prover on a regular basis, requiring the inputting of their various authentication factors to release the necessary secrets they put forward during enrollment. This is the primary idea behind using FFS as a vehicle for MFA.

ZKX IN DETAIL

ZKX is a novel, lightweight, multifactor authentication technology which verifies and authenticates user identities via fundamental ZKP algorithms including FFS and other similar protocols. Other popular ZKPs of identity are similar to FFS in their structure, but rely on different, computationally hard "one-way" problems to protect the secret data of users. For example, FFS relies on the difficulty of factoring modular squares, where the Schnorr protocol relies on the difficulty of the discrete logarithm problem.

ZKX utilizes a proprietary hashing method in order to derive functional ZKP secrets from traditional authenticators and identity evidence. Any repeatable hashing function can be used to accomplish this feat within ZKX. Artifacts rated from superior strength to fair strength (as outlined in NIST SP 800-63-3) are compatible within the ZKX architecture, and can be used to verify a claimant's identity. Authenticating secrets are split between what the human user will be expected to reproduce (e.g., password, PIN, RFID tag, etc.) and cryptographic data that resides on the device they have enrolled with. By themselves, these secret fragments are not enough to compromise any sensitive information about a Prover or their private authenticators. Authenticating secrets must be intentionally repopulated by the Prover through their release from another authenticator: whether it be physical, software-based, or in accordance with the principles of knowledge-based verification (KBV) like a PIN or a password. These key shares (the device-based and user-supplied data) are recombined, and injected with the per-round randomness native to ZKP algorithms (r, the round ID) which constitutes a Prover's response to a Verifier's challenge. Because of this structure, ZKX is resistant to breaches or leaks of information regarding authentication data. Even if the devices or both Prover and Verifier are compromised, there is no risk of revealing any authenticator data or identity-based evidence. This feature uniquely ties a user to the device they have enrolled with as a single "Prover", but can be adjusted to securely accommodate non-person entities (NPEs) as well.

Similarly, ZKX utilizes its proprietary hash expansion technique in order to derive multiple functional ZKP secrets from a single authenticating factor. This dramatically expands the number of secret keys, k, a Prover can be registered to use, further lowering the probability that an illegitimate user could successfully misrepresent themselves using ZKX. Because of this on-the-fly, multiple secret derivation from one distinct factor, Provers can be challenged using multiple keys while only having to perform one authenticating input. If their input is legitimate and honest, they will be able to correctly answer any number of Verifier challenges with ease. If an adversary is attempting to misrepresent their identity, they would need to successfully guess multiple authenticating values instead of just one. Because of its multi-factored nature, the probability that an adversary could cheat the ZKX system is exponentially reduced by its hash expansion.

The emerging trend of the Zero-Trust Architecture (ZTA) has encapsulated much of the strategic focus of the United States government and military for the foreseeable future. While ZKX is capable of securely and rigidly and seamlessly authenticating users in any organizational structure, it is designed with special attention given to the requirements and expectations of the mechanics of the ZTA. ZKX's form factor is that of a cryptographic software multi-factor authentication solution, a platform compatible with federal AAL3 requirements. ZKX is platform and communication agnostic, and is able to be employed using open standard technologies, allowing it to easily interoperate with other network components, if necessary via API or through a wrapper relationship. ZKX also includes a novel mechanism for users to attest their identities via ZKPs, enabling robust and constant zero-knowledge application session management. This session management is customizable to fit the requirements of local policies and can be revoked by an administrator if necessary.

ZKX is highly configurable and can be used to satisfy whatever requirements are applicable to the mission. ZKX can strictly and dynamically enforce local access policies regarding network resources, services, and platforms. Because ZKX is based on an arbitrarily repeatable proofing mechanism, Provers can repeatedly prove their knowledge or possession of the secrets tied to their unique identity until the Verifier is sufficiently satisfied. This can extend to the policy considerations an organization has in place to protect their resources. If a Prover must utilize multiple issued authentication artifacts in order to satisfy a given policy, more authentication rounds can be imposed on the user such that no doubt remains that they do indeed possess the necessary authenticators. Due to the nature of ZKPs, there is a measurable probability of assurance that an identity claimed by a Prover is legitimate. The probability than an adversary can successfully misrepresent their identity in ZKX's modified FFS scheme, for example, is 2^{-kt}. Where *t* is the number of ZKP rounds undertaken between Prover and Verifier, and *k* is the total number of user secrets (i.e., the length of vector **S**), which itself is the total number of authenticators a Prover could be expected to reproduce *(n)* multiplied by the number of functional secrets each authenticator is expanded into via the hash expansion process *(m)*. This probability is now calculable as 2^{-nmt}, and can be used to set thresholds for various "trust scores" which users must meet before they are able to access different resources.

For more information on ZKX, or a discussion with our Chief Technologist or Chief Scientist, please email info@zkxsolutions.com.