SplitCert
Passwordless Access to Databases

Bastion zero

Trust no one, not even us.

What is BastionZero's SplitCert?

BastionZero's SplitCert eliminates the operational overhead and security risk associated with maintaining, storing and distributing database passwords.

Passwordless Database Access

✓ **SplitCert provides a true zero trust database access** experience where there is no need to trust **anyone**, including BastionZero, with sensitive database passwords. Instead, the database authentication factor is split into two independent shards, and each shard is stored in an independent location. Then, whenever an authorized user needs access, the shards are used to construct short-lived database access credentials. Storing shards in independent locations allows SplitCert to eliminate the single point of compromise around database access and reduces the risk of data breach.

Invisibility

✓ **SplitCert is invisible to end users** and supports database access via popular existing database clients and workflows. BastionZero's initial release of SplitCert supports access to two popular databases: self-hosted Postgres and MongoDB.

Mutual TLS (mTLS) for Database Authentication

Mutual TLS is a method for mutual authentication with TLS. It allows a client and database to verify that the other party to the connection is who it claims to be by verifying that it holds the correct private key that corresponds with the public key in the presented TLS certificate. When mTLS is used for database authentication, the database client authenticates itself by presenting a client TLS certificate which is the child of a preconfigured, trusted certificate to the database.

How does it work?

- SplitCert uses Mutual TLS (mTLS) and cryptographic multiparty computation (MPC) to provide password-free authentication to databases.
- Each time an authorized user needs access to a database, SplitCert constructs a short-lived client certificate, on-the-fly, for mTLS authentication as the appropriately-scoped user on the database. (For example, **alice@example.com** logs in as the postgres user using a short-lived mTLS client certificate.).
- But, the unique aspect of SplitCert is that it avoids creating a single point of compromise. That's because the key used to construct the short-lived mTLS client certificates is not stored in a single location that could be compromised.
- Instead, SplitCert splits the database authentication factor into two shards. One shard is stored in the BastionZero cloud service while the other shard is stored in the customer's infrastructure. Although neither shard is sufficient to sign anything on its own, signatures from each can be combined using cryptographic MPC to construct the client certificate on the fly, without ever putting the two shards back together, and without ever creating a single point of compromise.

BastionZero's SplitCert MPC Protocol

BastionZero's SplitCert uses a form of MPC that allows a mTLS client certificate to be computed from two independent private-key shards held by two independent parties. One of the shards is stored in the BastionZero cloud, and the other shard is stored in the agent in the customer's environment. Storing shards in independent locations allows SplitCert to eliminate the single point of compromise. MPC is used to generate a short-lived mTLS client certificate from the two shards. The certificate that is then used to authenticate the user to the database.

BastionZero Agent

The BastionZero agent runs in the customer environment, on a server, container and kubernetes cluster that is "close to" the database. The agent should have the ability to create a network connection to the database (i.e., because the appropriate security groups and ACLs are established). If the database is self-hosted, then the agent may be deployed on the same server as the database.

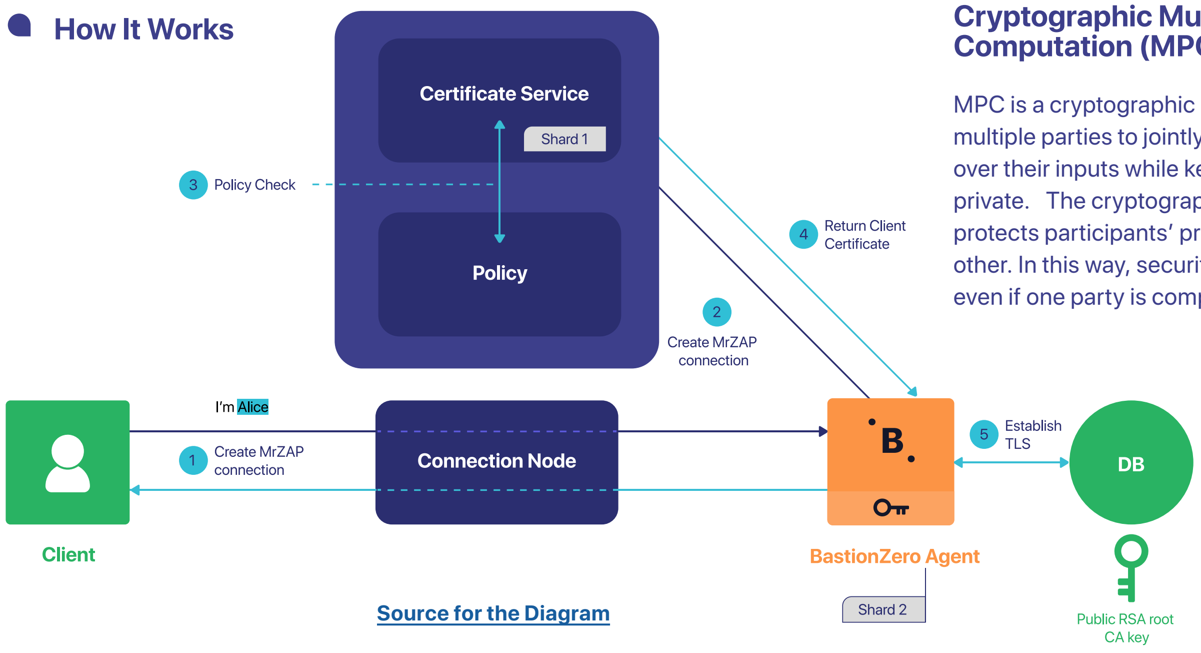
For more information about the BastionZero agent, see the BastionZero documentation.

If a user wishes to connect to a database, they must first pass through the usual BastionZero SSO, MFA, and policy check before connecting through the BastionZero cloud. Once those checks are completed, an authenticated channel is established between the end user and BastionZero agent in the customer's environment. **(See step (1)** in the figure below)

Next, the agent creates the contents of the certificate, and then initiates the MPC protocol used to generate a signature on the certificate. The MPC protocol involves the agent (which has one shard) and a certificate microservice in the BastionZero cloud (which has the other shard).

Once the MPC protocol completes and the certificate is fully signed, it is returned to the agent. Now, the agent presents the client certificate when establishing the TLS connection to the database **(See step (5))**. All traffic between the user and the database goes from client to agent over a BastionZero MrZAP authenticated channel, and then through the TLS connection from agent to database. Audit logging is performed at the connection node.

How It Works



Source for the Diagram

Cryptographic Multi Party Computation (MPC)

MPC is a cryptographic primitive that allows multiple parties to jointly compute a function over their inputs while keeping those inputs private. The cryptography in this model protects participants' privacy from each other. In this way, security is preserved even if one party is compromised.

The MPC Protocol

The MPC protocol used to create the signature is laid out in a technical paper from 2001 The Security of Practical Two-Party RSA Signature Schemes by Mihir Bellare and Ravi Sandhu.

Creating the shards.

The BastionZero cloud operates a certificate microservice that generates an RSA public-private key pair, where the private key is d. The RSA public key is certified by a root certificate and configured with the database. The certificate microservice will split the RSA private key, d, into two shards d1, d2 where:

d1 is chosen at random and
d1 + d2 = d [modulo the Euler totient of N]

The second shard d2 is sent to the BastionZero agent that will be used for access to the database. the original RSA secret key d is deleted. Splitting the secret into two parts, and storing one in the BastionZero cloud and the other in the customer's environment eliminates the certificate as a single point of compromise.

Signing a certificate on the fly.

To creates the contents of the certificate, the agent first chooses a fresh signing key pair (SK,PK) that it will use for the database mTLS connection. It then creates a mTLS client certificate for PK and then uses the RSA signature algorithm to sign the certificate using its shard d2, which produces a partial RSA signature sig2. The certificate is then sent over the BastionZero cloud service **(See step (2))** which uses the RSA signature algorithm to signs the certificate using its shard d2 producing the partial signature sig1.

Next, the agent multiplies the two partial signatures together to obtain the final signature as sig = sig1sig2 . The signature sig is the complete signature on the client certificate **(See step (4)).**

ABOUT

BastionZero

BastionZero delivers zero trust access to server, database, kubernetes and web infrastructure without creating a single point of compromise. It pairs with your IdP to quickly grant access with policy controls and observability — without a mess of passwords, VPNs, and SSH keys.

About the unique MrZAP protocol that BastionZero uses to create authenticated channels from client to agent, visit www.bastionzero.com

Learn more about BastionZero

RSAC Innovation Sandbox 2022 FINALIST