

CRYPTOSAT

YAN MICHALEVSKY, CO-FOUNDER

CRYPTO-SATELLITES

- A CRYPTOGRAPHIC TRUSTED PARTY IN SPACE AND
ITS APPLICATIONS



founders



Yonatan Winetraub

Aerospace



2nd time founder. Founded SpaceIL - the first private moon-mission.



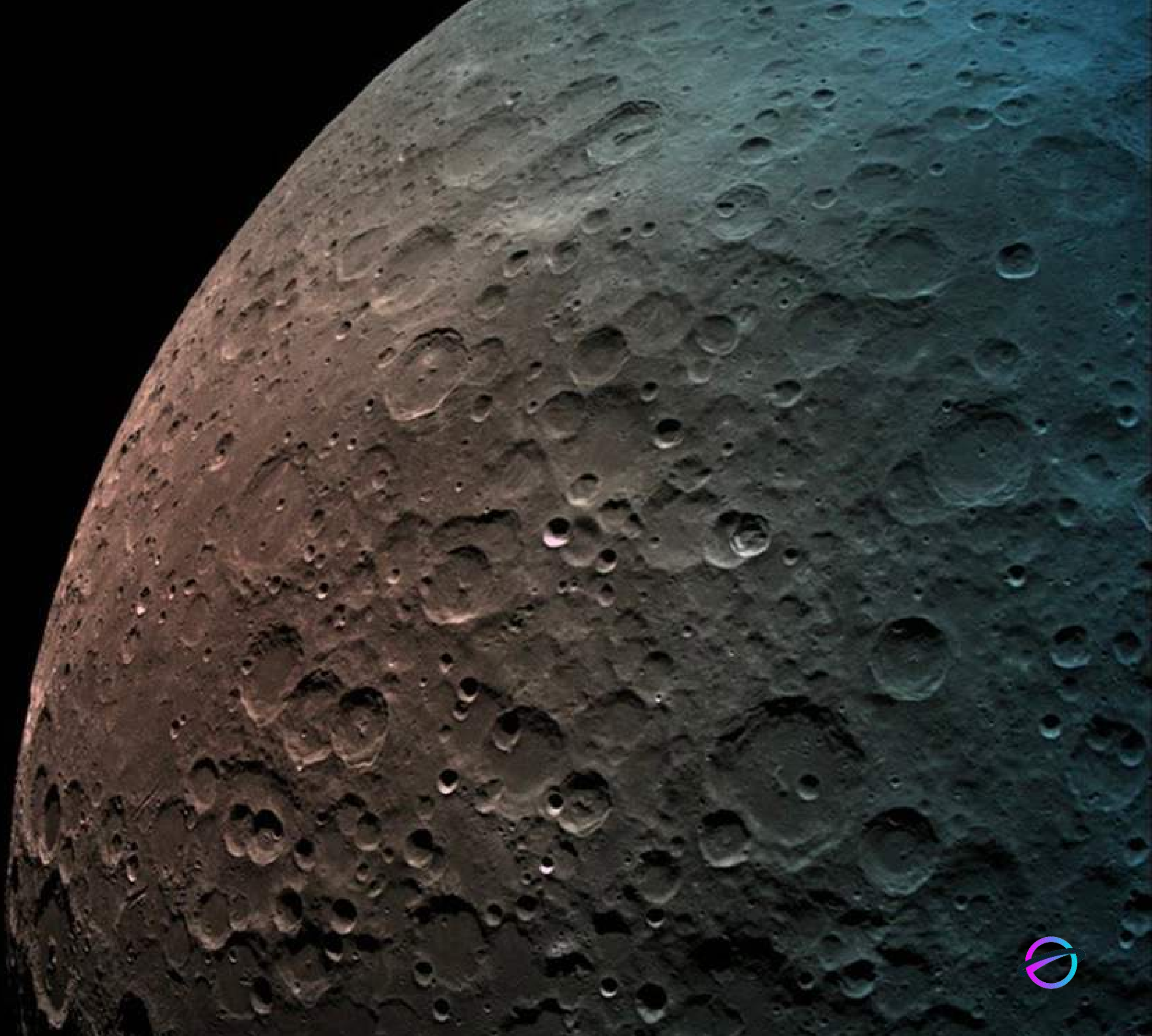
Yan Michalevsky

Crypto and Security



2nd time founder. Founded Anjuna - an enterprise security company in Confidential Computing.





Advisors



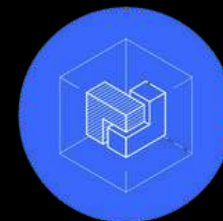
Prof. Dan Boneh



Ben Fisch



Juan Benet



Omer Shlomovits



mission

We build satellites that power cryptographic and blockchain applications. Our goal is to create the most secure cryptographic root-of-trust in space to protect protocols.



SpaceTEE

SpaceTEE: Secure and Tamper-Proof Computing in Space using CubeSats

Yan Michalevsky
Stanford University
yanm2@cs.stanford.edu

Yonatan Winetraub
Stanford University, SpaceIL
yonatanw1@stanford.edu

ABSTRACT

Sensitive computation often has to be performed in a trusted execution environment (TEE), which, in turn, requires tamper-proof hardware. If the computational fabric can be tampered with, we may no longer be able to trust the correctness of the computation. We study the (wild and crazy) idea of using computational platforms in space as a means to protect data from adversarial physical access. In this paper, we propose SpaceTEE – a practical implementation of this approach using low-cost nano-satellites called CubeSats. We study the constraints of such a platform, the cost of deployment, and discuss possible applications under those constraints. As a case study, we design a hardware security module solution (called SpaceHSM) and describe how it can be used to implement a root-of-trust for a certificate authority (CA).

However, via physical access to the HSM it may potentially be possible to obtain its secrets. HSM vendors such as Safenet and Thales incorporate many precautions and preventative measures to secure their hardware and make it tamper-proof. One defensive measure is to protect them with sensors that identify an attempt to open the HSM enclosure in order to access the board. The FIPS 140-2 US government standard [9] specifies the requirements for cryptographic modules that protect sensitive (but unclassified) information for commercial uses.

FIPS 140-2 specifies 4 levels of security. The specification for Level 4 states

"Physical security mechanisms provide a complete envelope of protection around the cryptographic module with the intent of detecting and responding to all unauthorized attempts at physical access."

RESEARCH-ARTICLE



WaC: SpaceTEE - Secure and Tamper-Proof Computing in Space using CubeSats

Authors: Yan Michalevsky, Yonatan Winetraub [Authors Info & Claims](#)

ASHES '17: Proceedings of the 2017 Workshop on Attacks and Solutions in Hardware Security • November 2017 • Pages 27–32 • <https://doi.org/10.1145/3139324.3139333>

Published: 03 November 2017 [Publication History](#)

[Check for updates](#)

0 98

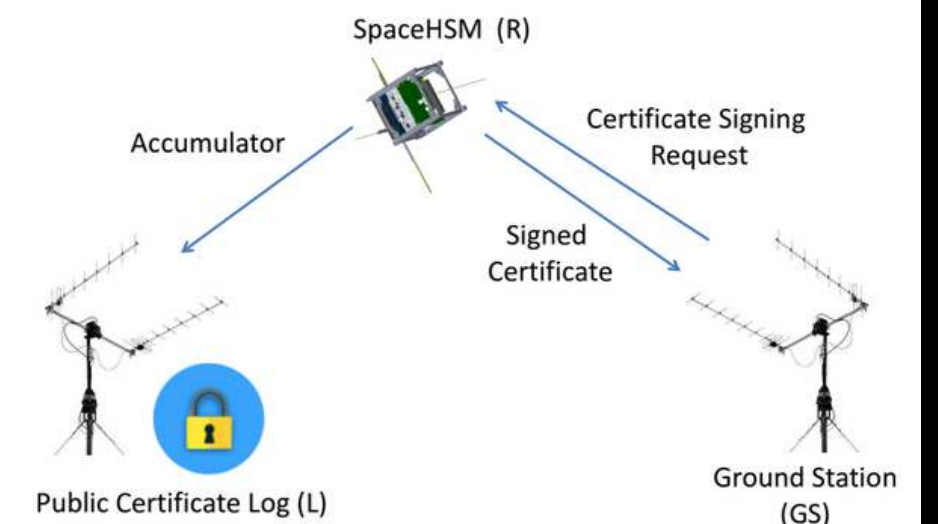
[Get Access](#)

ASHES '17: Proceedings of the 2017 Workshop...

ABSTRACT

Sensitive computation often has to be performed in a trusted execution environment (TEE), which, in

1. SpaceHSM: Root-of-Trust (R) and crypto-accumulator
2. Ground-station (GS): delegates cert. sign. requests
3. Public, append-only certificate log (L)
4. Verifier (V): checks certificate validity



Use-cases

HARDWARE SECURITY MODULE (HSM)

Can we provide a better alternative to expensive and hard-to-operate HSMs that can still potentially be physically accessed and might be susceptible to side-channel attacks?

TRUSTED SETUPS

How to generate public parameters for cryptographic schemes securely?
zkSNARKs, Treshold Signature schemes, etc.

TRUSTED SOURCE OF ENTROPY

How to obtain random bits we can trust to be random?

TRUSTED PARTY FOR CRYPTOGRAPHIC PROTOCOLS

Privacy-preserving aggregation, private voting, sealed-bid auctions

Why satellites are a great solution?

PHYSICALLY INACCESSIBLE & TAMPER EVIDENT

No side-channel attacks, a satellite can be destroyed but not captured. Any physical attack will be easily noticeable by national monitoring (e.g NORAD)

ZERO INFRASTRUCTURE

Self-sufficient, no internet cables, no power source, anybody can set a personal ground station

TRANSPARENT

Satellites can guarantee that multiple parties see the same transmission. No split-world attacks.

PRACTICAL

<\$100k to build, launch and maintain a satellite



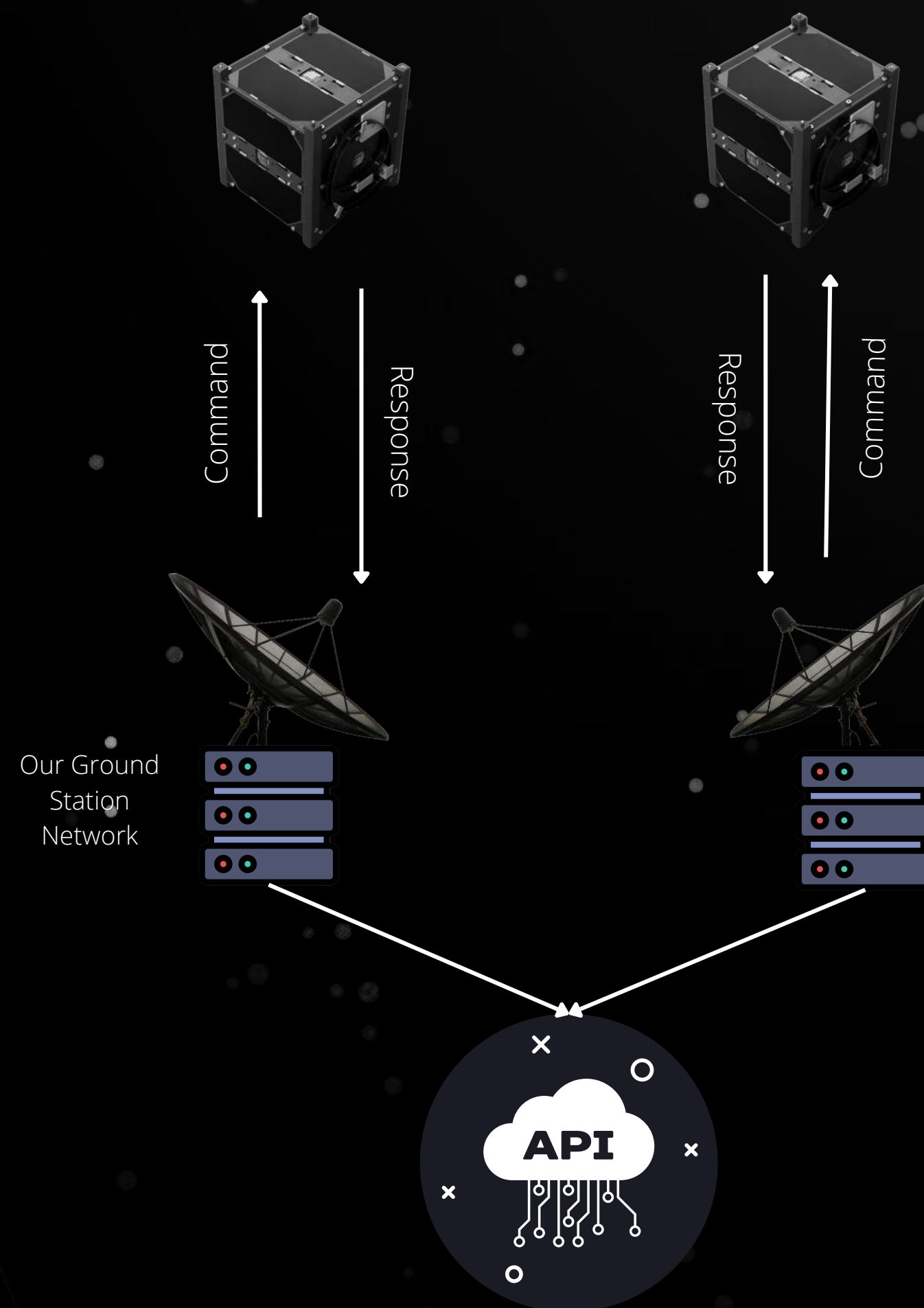
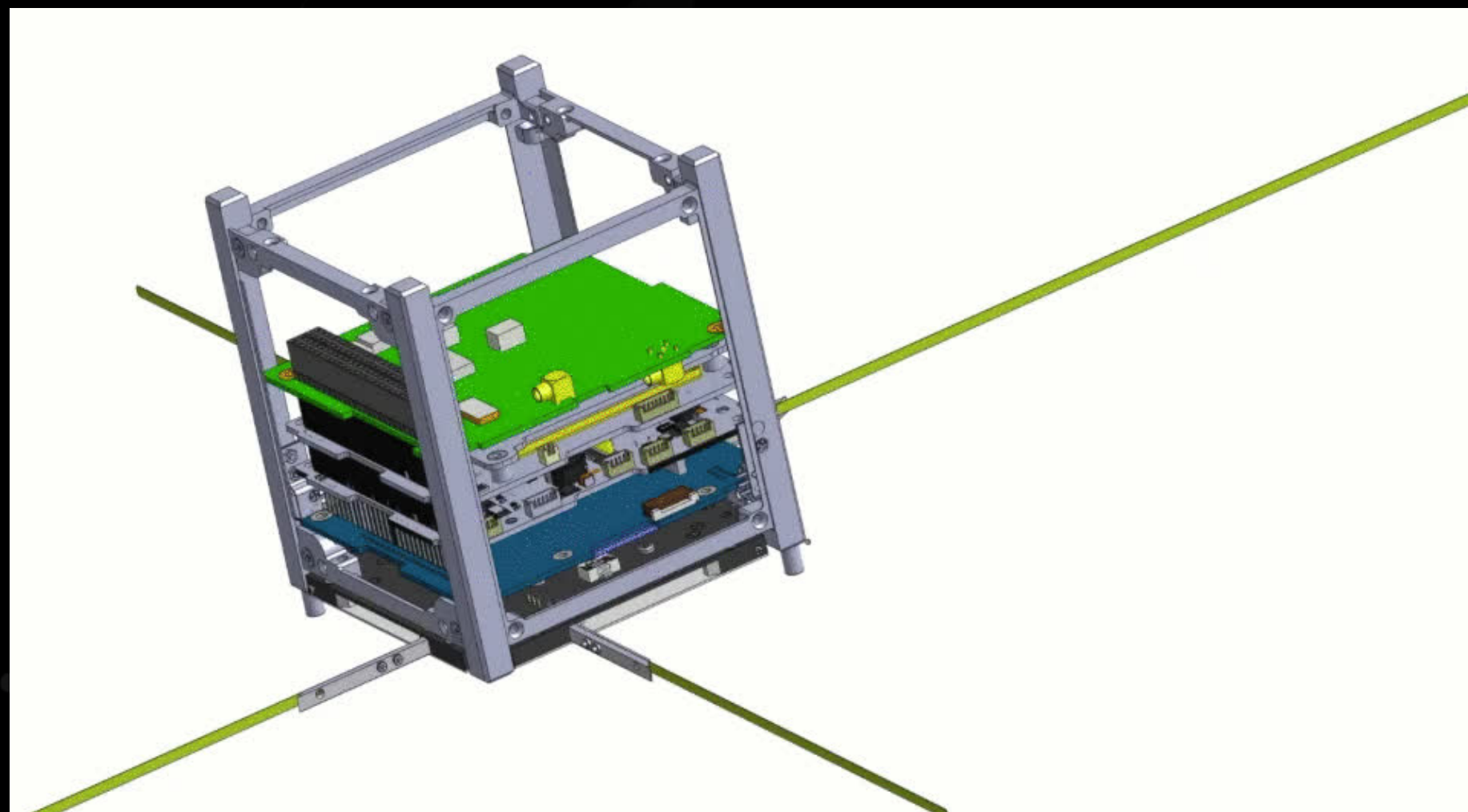
Cubesats

Cube satellites are miniature satellites made from off the shelf components. The costs of building cube satellites and launching them into space has dramatically declined in the past decades and will continue to do so as the technology gains more widespread adoption. We believe they are ideally suited for applications requiring trust.

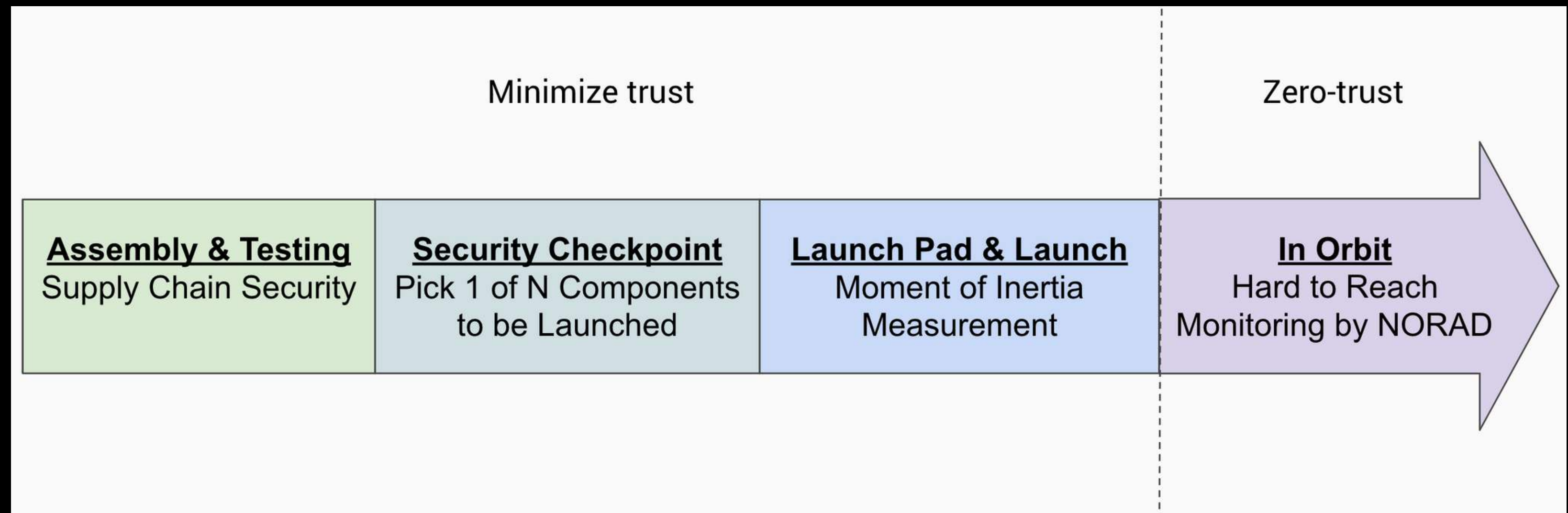


Architecture

(in a nutshell)



Trust model



Milestones



ISS Experiment
(March 2022)



Crypto1 Launch
(May 2022)



2nd ISS
Experiment
(December 2022)



Crypto2 Launch
(Jan 2023)
30x compute power

Crypto3
In Design



1st ISS Experiment

Cryptosat experiment aboard the International Space Station (ISS).

- DRAND (Distributed Random Beacon)
- Bitcoin transaction co-signing from space
- Tweets From Space
- and more...



Historic Experiment at ISS Takes Blockchain Technology to Space

One of your neighbors posted in Neighbor News. Click through to read what they have to say. (The views expressed in this post are the author's own.)

P Palo Alto / Mar 22, 2022

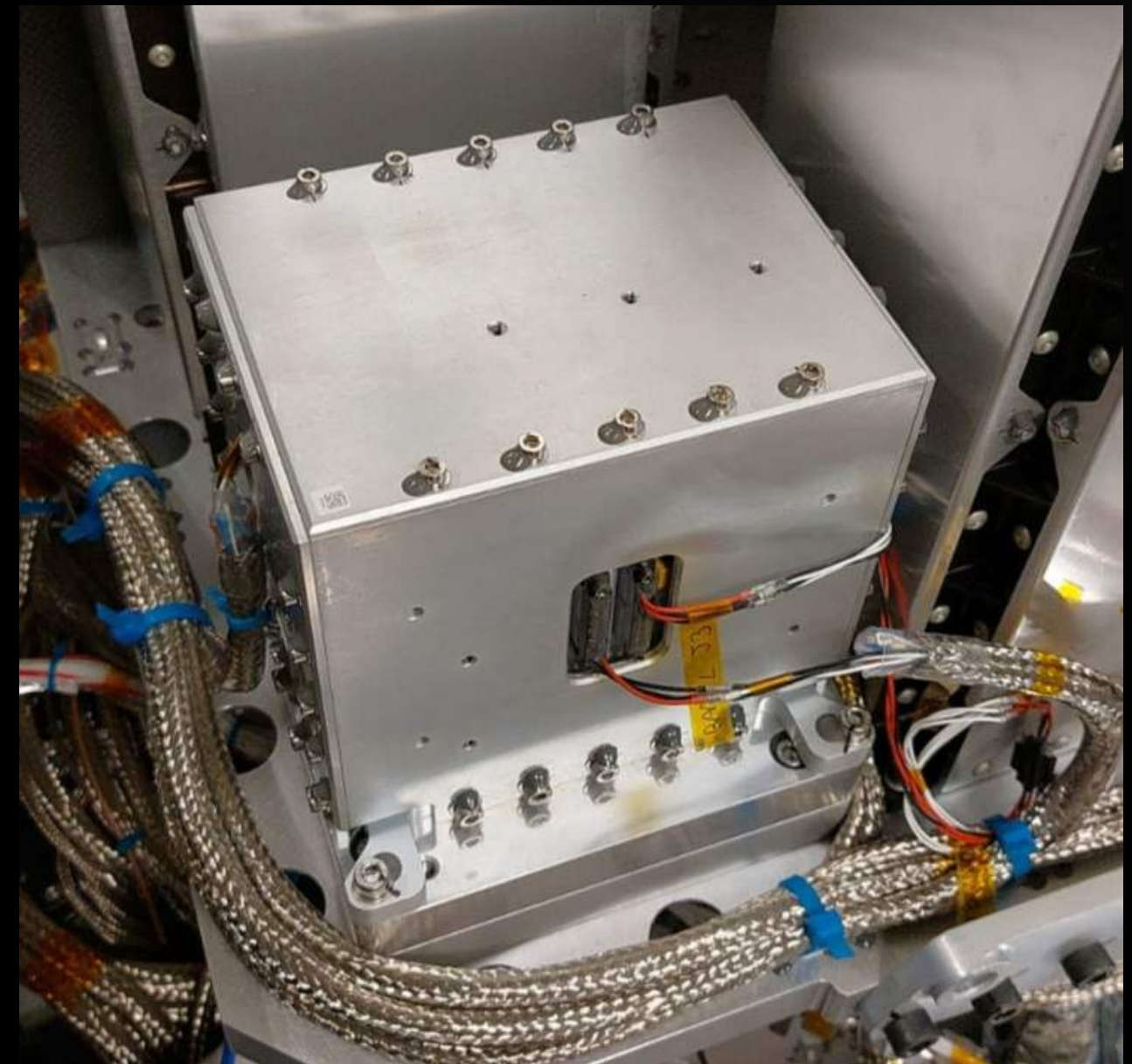
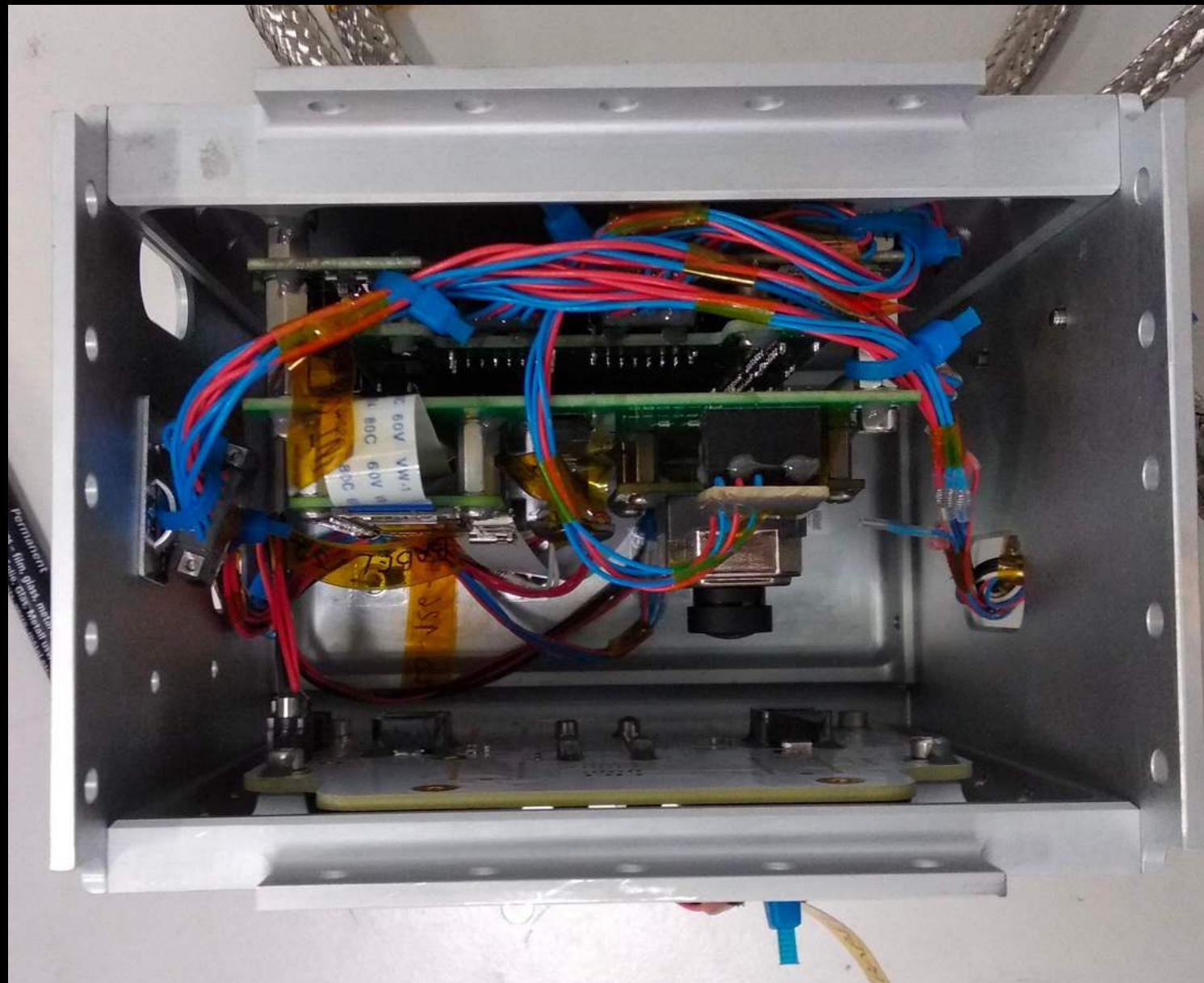
Tweets From Space

GET YOUR TWEET SIGNED ABOARD THE INTERNATIONAL SPACE STATION (ISS)

Tweet at [@cryptosatbot](https://twitter.com/cryptosatbot), and we send back a certificate digitally signed in space.



Crypto1 and Crypto2



Crypto1 launch, May 25th 2022



Trusted Setup

TRUSTED SETUP IN SPACE FOR
GROTH16 ZK-SNARKS

The proof system is used for collusion resistant voting in a DAO.

TECH • JANUARY 12, 2023, 9:00AM EST

Crypto in space: Cryptosat and DoraHacks complete ZK proof experiment on space station

by [Jeremy Nation](#)



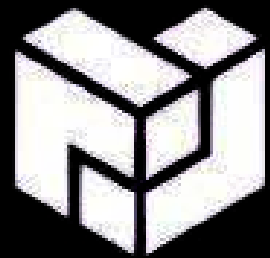
Crypto2 launch

Cape Canaveral, Florida

Jan 3rd 2023



Some of our partners



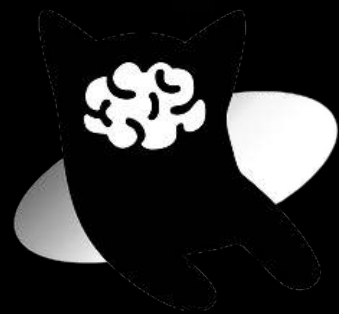
Protocol Labs



ethereum



drand



DoraHacks



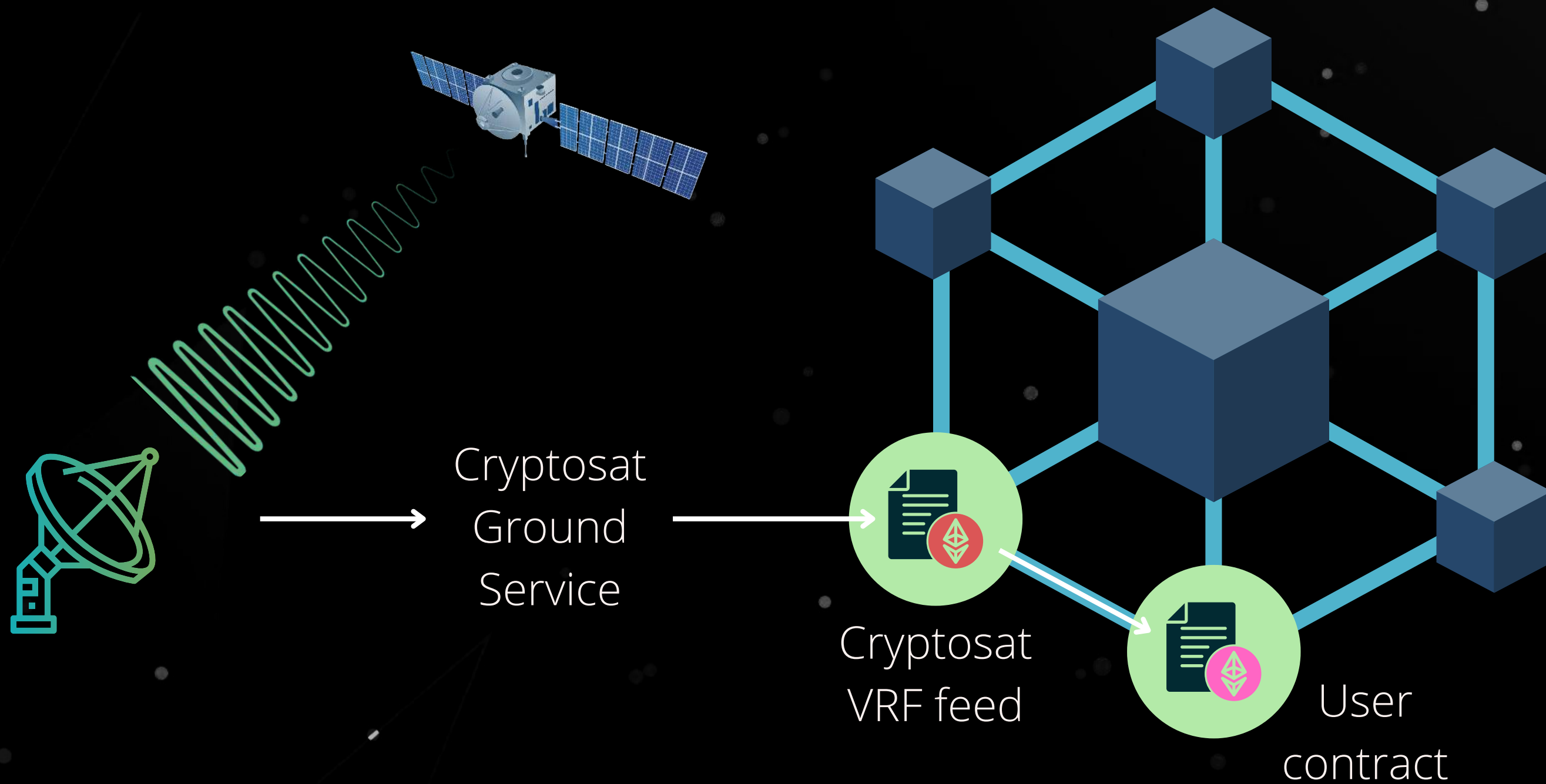
VELAS



MPC
ALLIANCE



CRYPTOSAT VRF ORACLE



CRYPTOSAT RANDOM BEA

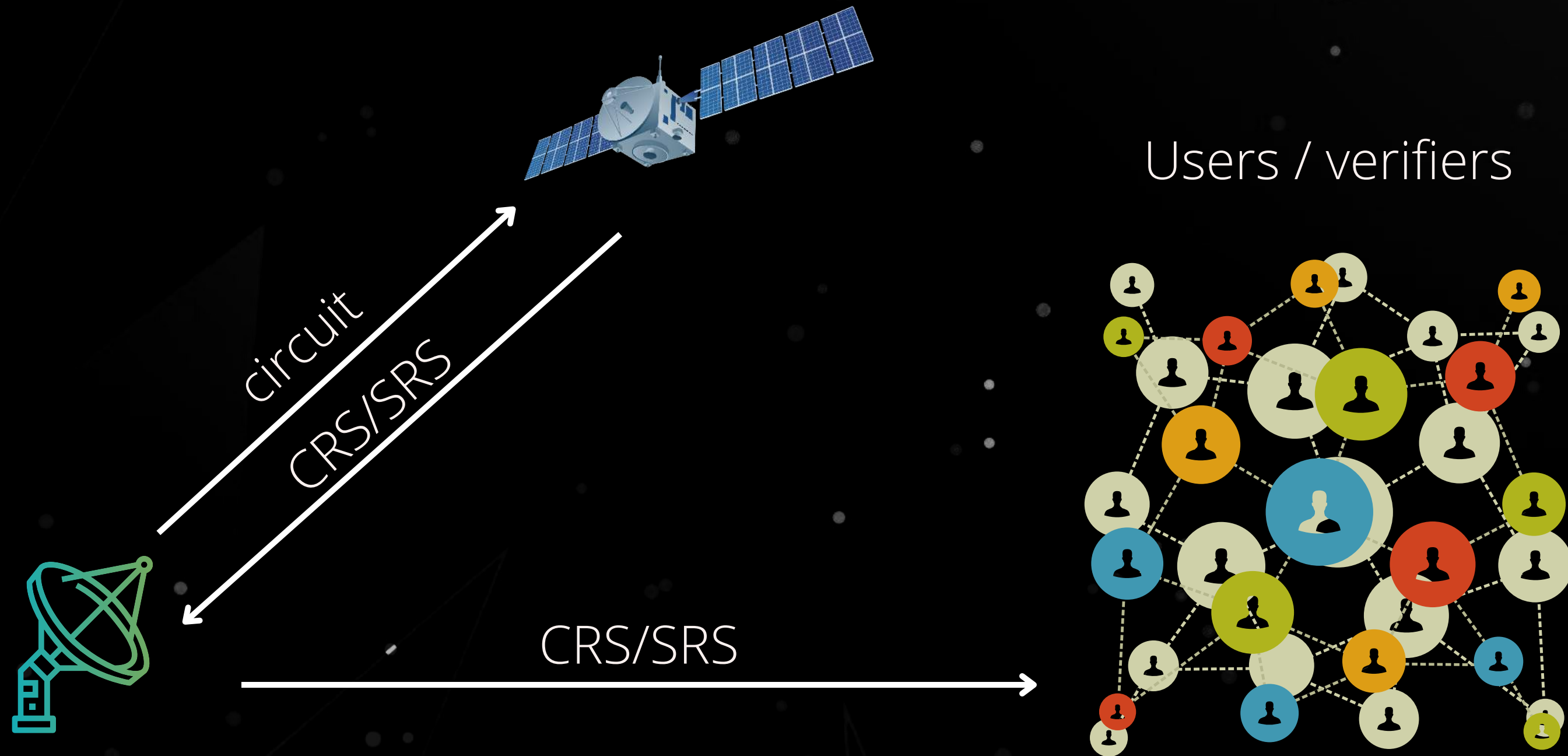
The screenshot displays a web interface for deploying and running transactions on a blockchain. The interface is split into several panels:

- DEPLOY & RUN TRANSACTIONS:** This panel on the left contains settings for the deployment environment, including the network (Injected Web3), account (0xFd0...2cBC7), gas limit (3000000), and value (0 Wei). It also shows the contract name (ClientContract) and a 'Deploy' button.
- Code Editor:** The central panel shows the Solidity code for the 'ClientContract'. The code includes imports for 'SpaceRandomBeaconInterface.sol' and 'SignatureVerifier.sol', a constructor that initializes the beacon, and a 'getRandom()' function that calls the beacon's 'getRandom()' method and verifies the signature using 'SignatureVerifier.verify()'. The function returns the signer, randomness, and signature.
- Debug Console:** The bottom right panel shows a call to 'signature()' from the 'ClientContract' to the 'signature()' method. The input data is '0x51f...f4847'. The decoded output is a JSON object:

```
{ "0": "bytes: 0x40017b8c8a5f1c16e8ddb6774c69d7330f8e4ar53c8ad0b40c7429ba19213f092f2f3a0f08307fb18c3dc0a8d2cdee311c" }
```

<https://docs.cryptosat.io/cryptosat/random-beacon/verifiable-random-beacon>

Trusted Setups for zkSNARKs



How Developers Interact with Cryptosat

CRYPTOSAT SIMULATOR

Private Randomness

The cryptosat can serve private randomness to users. Users can supply their own public key and the cryptosat service will generate random bits and encrypt them with the user-provided key. The cryptosat also signs the message using its signature key.

To obtain private bits from the cryptosat first generate a local key pair and a nonce:

```
clientKey = nacl.box.keyPair();
nonce = nacl.randomBytes(nacl.box.nonceLength);
```

Then invoke the Cryptosat API call:

```
request = cryptosat.getPrivateRandom(clientKey.publicKey, nonce);
```

The API call returns a request object allowing the user to track its status. The status can be obtained by invoking the status method:

```
request.status();
```

After the cryptosat signed the message and transmits the signature back to earth, the status of the message will change to *Ready* and the result of the request can be obtained by invoking the result method:

```
result = request.result();
```

You can then decrypt the message using the following snippet:

```
encryptionKey = cryptosat.getPublicEncryptionKey();
plain = nacl.box.open(result.encryptedRandom, nonce, encryptionKey, clien
```

And verify that the signature is valid using the this snippet:

```
clientKey = nacl.box.keyPair();
nonce = nacl.randomBytes(nacl.box.nonceLength);

< Uint8Array { ... }

> request = cryptosat.getPrivateRandom(clientKey.publicKey, nonce);

< Request { ... }

> request.status();

< "Ready"

> result = request.result();

< Object { encryptedRandom: Uint8Array, signature: Uint8Array }

> encryptionKey = cryptosat.getPublicEncryptionKey();
plain = nacl.box.open(result.encryptedRandom, nonce, encryptionKey,
clientKey.secretKey);

< Uint8Array { ... }

> signingKey = cryptosat.getPublicSigningKey();
nacl.sign.detached.verify(result.encryptedRandom, result.signature, signingKey)

< true

> |
```



Thank you

IN THE NEWS

[Space Invaders](#)

[TheBlock](#)

[Crypto1 launch](#)

[Yahoo finance](#)

PL + CRYPTOSAT

[VDF announcement](#)

VELAS + CRYPTOSAT

[Random Beacon announcement](#)

WHITEPAPER

[SpaceTEE: Secure and Tamper-proof Computing in Space using Cubesats](#)

