# Strong Builder Identity

co authored with @kobigurk from Geometry and @mmp from SMG

Thanks to @akinovak for comments.

5 Hours ago, Titan Builder announced on twitter that they had evidence of a builder impersonating them. In the post Titan described how an imposter has been using Titan's extradata, "Titan (titanbuilder.xyz)", in blocks produced by the imposter. The imposter also set Titan's address as the block recipeint (titanbuilder.eth). The only identifying information that the attacker couldn't spoof was the proposer payment transaction which it sends from (titanpayload.eth).

You can see a recent example imposter block here.

What the imposter is trying to acomplish is unclear at this stage. That said, as of the time of writing Titan estimates they have spent over 17 eth and an attacker rarely does this without a plan for making it back.

There are also possibly connected reports of a flashbots imposter.

Motivated by this, we suggest a stronger builder identity system. Rather than rely on plaintext extradata to identify builders, which is easy to spoof, the extradata should be used for a builder signature.

The main design constraint we have is that the system must integrate into the current block structure.

## Components

### BN254 BLS signatures

BN254 is a pairing-friendly curve commonly used for SNARKs, and has cryptographic operations precompiled on Ethereum since EIPs 196 and 197.

BLS signatures are a pairing-based signature scheme, where the signature is a single group element. For BN254, this means we can encode a signature in 32 bytes.

Additionally, it's possible to (somewhat) efficiently verify BLS signatures on Ethereum.

### Registry

The registry smart contract would accept the following from builders:

- Builder info:
- Name
- BN254 public key
- .eth address and a signature from it (or some other identifiying mechanism)
- Name
- BN254 public key
- .eth address and a signature from it (or some other identifiying mechanism)
- BN254 signature on the builder info hash

The registry would verify the signature before approving the builder.

The builder info would be stored in an append-only list, and the builder would keep their index in this list, denoting it builder index

.

### Graffiti

The builder has the ability to set the extra data

field in a block, which is 32 bytes long. The builder would put a BLS signature on the block hash in the extra data.

Proposers may verify the signatures before proposing, but they don't have to.

## Gas limit watermarking

The builder also has to tell the signature verifiers which builder they are, and they don’t have enough space in the extra data anymore.

Note that the builder can at least control the last 3 digits in the gas limit arbitrarily, allowing them to use it for their builder index.

## Verifiers

Verifiers would then get the corresponding name and public key from the smart contract.

Explorers can use this data to verifiably display builder identities in blocks.

## Limitations

Note that with current relay design, the relay can change the gas limit unilaterally, which may be a concern. However, a malicious relay can already do a lot worse things than this in the present design, so we are comfortable assuming a trusted relay.

## Other design possibilities

- Use longer but cheaper signatures and encode them in the proposer payment transfer transaction. The problem is that the proposer payment transfer txn does not always exist
- Encode the builder index in the same place. Same problem