2.1 Address-Based Encryption

We propose a variant on identity-based encryption, called address-based encryption. Rather than directly using an e-mail address or phone number as a public key, and then relying on a trusted private-key generator to generate a corresponding private key, we have users generate their own private/public key pair in the traditional manner. The user then registers their public key in a public, append-only database that stores `[address -> public key]` tuples. This database is functionally decentralized, so that no central owner is responsible for storing, managing, or maintaining the database, but logically unified, so that everybody can at any time see all the entries in the database. Crucially, the `[address -> public key]` tuples are attested to by a peer-to-peer network. To perform attestation, randomly selected validators in the network send a signed and secure message to the registrant, who then signs the message with her private key and returns it to an Attestations smart contract. The Attestations contract checks that the validator did indeed send the message, and that the signature matches the public key of the recipient.

This protocol works not just with email addresses, but with any channel to which a secure message can be sent, for example, a cell phone number, an IP address, or even a bank routing and account number. Further, arbitrary strings may be appended to the address in the database key, allowing multiple public keys to be stored for each address, each for a different application. As a consequence, the encryption scheme supports a large number of cryptographic applications, from two-factor authentication to decentralized social networks, without relying on trusted third-parties.

For the social payments use case, it allows for two important features. First, a user can send Celo currencies to a friend by using her phone number as the public key, allowing easy payments to contacts. Second, a user can send Celo currencies to a friend even if the friend has not yet downloaded a Celo wallet.

2.1.1 Single-Node Address-Based Encryption

For the purposes of explanation, we begin by describing a simplified version of the address-based encryption scheme in which a single node, called a validator node, maintains the state of the system.

The key role of the validator node is to maintain a public, append-only database of `[address -> public key]` mappings. In the single node case, the validator node is similar to a traditional key server except that it not only stores the `[address -> public key]` mappings, but also attests to them as follows:

When a user wishes to register a public key with the scheme, they generate a private/public key pair, and then submit their `[address -> public key]` mapping to the validator node. (In our use case, the address is the cell phone number of the user, but in the general case it could be any address to which a secret message can be sent.) The validator node sends a signed secret message to the address in the entry. The user then sends that message to an Attestations smart contract, which verifies both signatures by decrypting them with the public keys of the user and the validator. If the decrypted message matches the secret message, the smart contract writes the following entry to the database `[address, user public key, secret message, user signed secret message, validator signature]`.

2.1.2 Drawbacks

This simplified version has the following drawbacks:

Address harvesting. A publicly viewable database with unencrypted phone numbers allows spammers to harvest the cell phone numbers of all of the users. To address this, we can store a one-way hash of the address rather than the address itself. To increase the entropy of the underlying string (to make reversing the hash more difficult) we may append a pepper to the string to be hashed[2].

[2]Even with an appended pepper, the following scenario is possible: a spammer one-way hashes every possible 10 digit number along with every possible pepper, and then checks to see which hashed values are in the database. However, harvesting at high cost is possible even today, by taking every possible 10 digit number, sending an SMS to each, and seeing if it goes through. Therefore, our goal would be to make the effective cost of decryption more expensive than the cost of sending a bulk SMS.