Complete Knowledge

IC3

Follow

The Initiative for CryptoCurrencies and Contracts (IC3)

--

1

Listen

Share

by James Austgen, Kushal Babel, Vitalik Buterin, Phil Daian, Ari Juels and Mahimna Kelkar

‘In this life, we want nothing but Facts, sir; nothing but Facts!’

— Charles Dickens, Hard Times

In a paper we’ve released today, we introduce a new cryptographic notion that we call proofs of complete knowledge

(CK). We also report on a prototype that offers a path to making CK practical for use with smartphones.

Proofs of knowledge, as their name suggests, enable one party, e.g., Alice, to convince another party, e.g., Bob, that she knows a particular secret — such as a private key SK

. Digital signatures are an example. Consider a cryptocurrency transaction T

created by Alice that spends funds from her address A

. To be valid, T

must be digitally signed with the valid private key SK

for A

. The digital signature here proves that Alice, as creator of T

, knows the key SK

. It’s this proof of knowledge of SK

that makes the transaction valid: In a cryptocurrency system, the owner of address A

is by definition

whoever knows SK

.

Proofs of knowledge are a mainstay of cryptographic system design. It turns out , however, that existing proof-of-knowledge systems have a serious limitation: Alice may be able to create a convincing proof that she knows SK

, yet she may still not truly

know it.

Here’s an analogy. Suppose Mallory wants

to appear as a contestant on Jeopardy

. She does an interview with the show’s host (Mayim Bialik). Mallory knows that she’ll get questions during the interview about U.S. state capitals, which she never learned in school. (She was busy flipping NFTs during geography lessons.) So she decides to cheat. Mallory’s friend Alice knows all of the state capitals in the eastern half of the U.S., while her friend Bob knows the ones in the west. Mallory therefore hides a wireless communication device in each ear. The one in her left ear communicates with Alice. The one in her right ear communicates with Bob. Thanks to Mallory’s deception, when she’s asked the capitals of various U.S. states during her interview, she always answers correctly.

But does Mallory know all of the U.S. state capitals? Clearly not. Does Alice or

Bob? The answer is still no. We can perhaps say that Alice and Bob together

know all U.S. state capitals. But suppose that Alice is strangely superstitious and refuses to answer state-capital questions on Fridays. Can we still say that Alice and Bob together have knowledge of all U.S. state capitals? Yes and no. The knowledge exists in Alice and Bob's heads, but it's not necessarily possible to extract it as desired

. If Mallory does her Jeopardy

interview on a Friday, she's in trouble.

Complete knowledge addresses a gap of this kind in what it means to have knowledge. In our analogy, Alice's refusal to answer questions on a Friday is a restriction in access to knowledge that we refer to as encumbrance.

To explain where existing proofs of knowledge in cryptography fall short, it's helpful to start with some potentially problematic examples of encumbrance of secret keys in particular.

The Encumbrance Problem

There are two different technologies that can encumber secrets: trusted execution environments

(TEEs) and secure multiparty computation

(MPC).

TEEs are computational environments that resemble black boxes: Their internal state is opaque and tamper-resistant. Using either hardware or software, they enforce both the integrity

and the privacy

of executing applications, providing protections against tampering and observation even by the owner of the very machine on which the code is running. Examples of TEEs include enclaves (e.g., ARM Trustzone) on mobile devices and Intel Software Guard eXtensions (SGX), which is available on some cloud servers.

On the other hand, MPC is an algorithmic approach to achieving much the same security goals. It splits secrets and computation among a collection of servers often called a committee. The security of MPC relies on a quorum of servers in the committee (e.g., a majority) behaving correctly.

As TEEs are much more practical and widespread than MPC today, we'll focus on TEEs in the following examples, starting with a TEE-based attack on blockchain voting systems. Here's an example involving an attack on voting schemes.

The problem of strong, stealthy bribery in this example affects not just naively designed blockchain voting schemes, but even ones specifically designed to resist bribery, such as the Minimal Anti-Collusion Infrastructure (MACI).

Even worse, a TEE can stealthily incentivize and grow a community of users mounting a unified attack against a protocol — a concept we have previously dubbed a Dark DAO. Such attacks could involve not just tampering with votes but manipulating asset prices and possibly attacking off-chain systems.

What's particularly pernicious is that if Alice sends a signed transaction to a smart contract using an encumbered key SK

, there's no way for the contract to tell. Alice's transaction will look identical

to one created with an unencumbered key. So it seems that it's simply impossible to prevent bribery attacks of the kind illustrated in our example.

But is it really?

Ensuring Complete Knowledge

It turns out that there is

a way to ensure that a key SK

is unencumbered — specifically that somebody

has direct access to SK

. Informally, this is what we mean by Complete Knowledge. The effect of CK is that bribery schemes like that in our example won't work, because there's no way for a malicious user like Mallory to prevent a user from using her key however she chooses.

There are two basic approaches to enforcing CK. Both enable generation of what we call proofs of complete knowledge

(PoCK), and specifically ones that are zero-knowledge

, i.e., don't reveal SK:

1. The fight-fire-with-fire

approach

Suppose that RBU adds a twist to its referendum. In order to participate, a voter like Alice must use a voting key SK

that's already in a TEE application created by RBU.

This application does only two things: (1) It generates a PoCK, i.e., proves to RBU that a particular private key SK

is in the application and (2) It spits out the key SK

to the user on demand. The result is a guarantee of complete knowledge of SK

by Alice. Now, Mallory has no way of encumbering Alice's voting key SK

in a TEE, because it's already in a TEE application

.[1]

Most users have TEEs on their mobile handsets. We show in our paper how they can use them in a CK scheme for voting or any other application.

As we discuss in our paper, however, there are some technical limitations to TEEs, particularly the ones in mobile handsets. So we also consider a second approach.

1. A proof-of-work-based

approach

This approach is a slightly counterintuitive one. It involves using a mining ASIC, i.e., a proof-of-work mining hardware rig, such as the kind used for Bitcoin. ASICs have two nice properties for the purposes of CK. They can perform certain computations, namely hashing

, far faster than ordinary CPUs or GPUs — on the order of one million times faster than a state-of-the-art CPU! And they do not include TEEs.[2]

Drawing on and modifying techniques from literature on the theory of proofs of knowledge, we have designed a protocol that involves use of an ASIC for mining in a way that involves SK

(yet cleverly hides it). This protocol generates a proof that SK

has been sent to the ASIC, i.e., a PoCK. Since the ASIC doesn't have a TEE, this means that SK

is effectively exposed to the ASIC user. This property is essentially equivalent to what's done by RBU's special TEE application: spitting out SK

directly to the user. Because the ASIC is so much faster than platforms that support TEEs, there's no way to generate an equivalent PoCK using a TEE.

Our paper shows experimentally that even using a cheap (roughly $200), outmoded (circa 2017) ASIC, it's possible to implement a high-security PoCK.

Applications of CK Beyond Bribery-Resistance

While we designed CK initially to address the threat of bribery in voting schemes, there are many other applications. CK can help prevent encumbrance of decentralized identity credentials, e.g., attestation that a user has undergone anti-money-laundering diligence. This is essential to prevent the many kinds of mischief possible if users can rent out their credentials, e.g., an identity-credential market for bypassing sanctions.

CK also has interesting applications in NFT marketplaces. For example, it can help ensure enforcement of royalty payments for artists. Today, royalties are only enforced off-chain, i.e., voluntarily

, by NFT marketplaces. They could instead in principle be enforced on-chain

in a smart contract, in which case enforcement would be mandatory. One reason mandatory royalty payments aren't considered good practice is that it would be unfair for users transferring NFTs between their own addresses to have to pay royalties. The problem is that there's no way to distinguish between a sender address S

and receiver address R

belonging to a single user and addresses S

and R

belonging to distinct users. Thus there's no way to distinguish between a single-user transfer and a change of ownership of an NFT.

With CK, though, it is

possible to distinguish. An NFT marketplace contract could require for royalty-free transfers that a PoCK be applied simultaneously

to R

and S

, proving that a single entity knows both

. This property is a pretty good approximation to single ownership. At least we believe that this CK-based approach could strongly disincentivize royalty evasion.[3]

Another interesting application of CK to NFTs is a notion we callAtomic NFTs. It is common to fractionalize

expensive NFTs, that is, to subdivide their ownership into small pieces, as happened for instance with theDoge NFT.

A number of artists we've spoken with, however, prefer that their works be owned and appreciated by a single individual. Atomic NFTs offer a way to create NFTs that respect this desire and ensure sole ownership. To demonstrate this capabilities, we've minted our own small collection of Atomic NFTs, shown here.

Conclusion

Proofs of knowledge

are a well studied, decades-old concept in cryptography. But powerful new(ish) technologies such as TEEs and MPC threaten to upend the classical proof-of-knowledge landscape through encumbrance

of secrets. CK offers a way to prevent such encumbrance and restore the intuitive idea of knowledge being attributable to a single entity or individual. In addition to offering a remedy to encumbrance-based attacks — such as bribery in voting schemes — it also offers a range of new applications.

We believe CK can be practical. We've implemented an end-to-end ASIC-based system with asmart contract that verifies PoCK proofs. Our prototype system includes a smart-contract-based registry of addresses proven to be CK compliant. We've also implemented a prototype system using mobile-handset-based TEEs. Our prototypes aren't yet ready for prime time, but we believe there's a path to practical, production deployment. You can find an open-source version of our tools here. We welcome collaboration from projects interested in realizing it.

[1] Happily, TEEs can't be nested, so there's no clever way for Mallory to wrap RBU's TEE application and thereby still encumber users' keys.

[2] Most ASICs have very little onboard security. They don't even support encrypted networking, as we discovered when our ASIC was apparently compromised by a remote hacker several times in the course of our experiments.

[3] There is a caveat here. As we discuss in our paper, if social recovery mechanisms become commonplace, CK proofs of simultaneous knowledge of keys would no longer work.