Programmable Cryptography (Part 1)

by gubsheep. Thanks to Justin Glibert, Barry Whitehat, Vitalik Buterin, Yan Zhang, Alex Allain, Dan Boneh, Elaine Shi, Nicholas Brysiewicz, and Albert Ni for discussion and review.

Cryptography is undergoing a generational transition, from **special-purpose cryptography** to **programmable cryptography**.

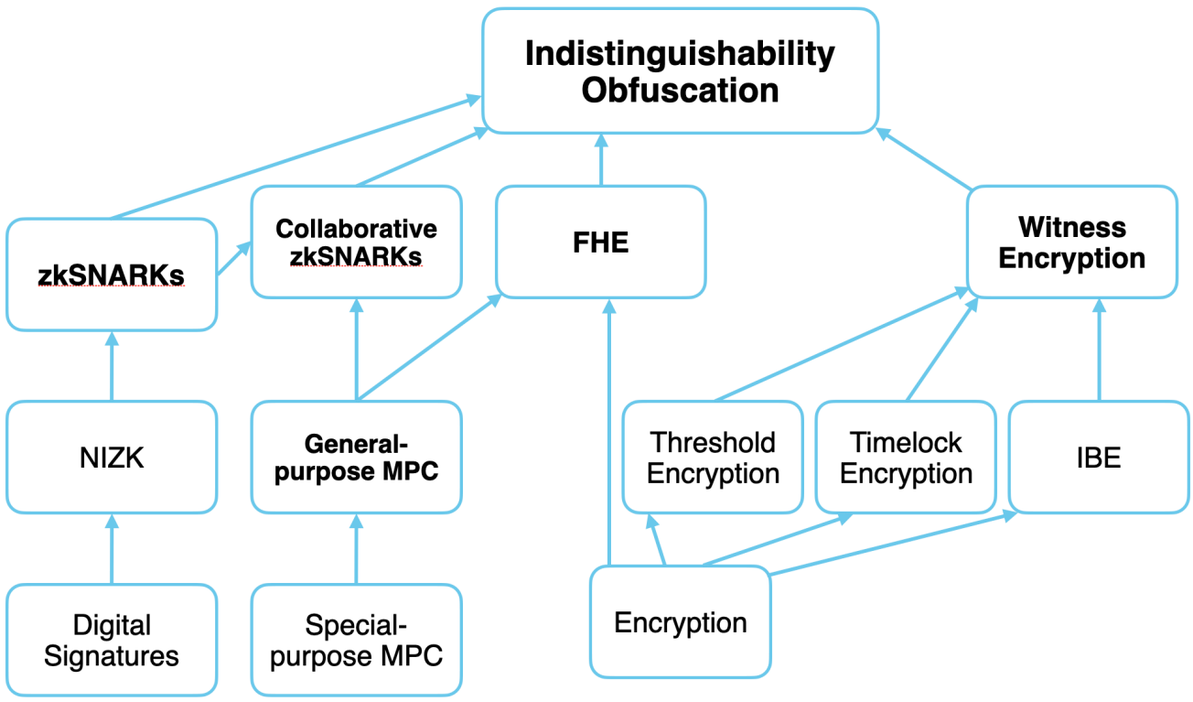
By "special-purpose cryptography," we mean protocols that allow us to perform *a single operation* with cryptographic guarantees. Public-key encryption and signature schemes are examples of special-purpose cryptography: for example, a signature scheme allows me to prove to you that I know a unique cryptographic key. More complex examples of special-purpose cryptography include group signatures, a special type of signature scheme which enables one person to sign anonymously on behalf of a group, or range proofs, which enable you to prove that a secret number is within some range, without revealing the number. In each of these cases, cryptographers have designed a special-purpose protocol, sometimes even inventing new math, in order to enable one specific operation to be performed with cryptographic guarantees.

Over the past 50 years, special-purpose cryptography has become an irreplaceable component in the global communications stack—for example, almost all data transmitted on the Web today passes through cryptographic protocols of some form.

We use the term "programmable cryptography" to refer to a second generation of cryptographic primitives that are becoming practical today. The defining feature of these primitives is that they are far more flexible than first-generation cryptography: they allow us to perform general-purpose computation inside or on top of cryptographic protocols. Here are a few examples:

- **Fully-homomorphic encryption** (FHE) allows me to compute an arbitrary program over someone else's private data, without learning anything about the data or the output of the computation.
- **Multi-party computation** (MPC) allows me to compute an arbitrary program over multiple peoples' private data, without learning anything about the data *except for the output of the computation*.
- **zkSNARKs** allow me to prove that an arbitrary piece of code was executed correctly on some secret inputs that only I know, without revealing anything

about those inputs.
    - With "special-purpose" zero-knowledge proofs, I could make a claim like: I have a secret S such that "MODEXP(S) = 0xa91af3ac..." (where MODEXP is the modular exponentiation function), and I can prove this to you without revealing S.
    - With "general-purpose" zkSNARKs, I can do this not only for the specific function MODEXP--I can do this for any function you want.
- **Witness encryption** allows me to encrypt a message to a *program*, rather than a secret key. For example, I could set up a math research prize by encrypting a Bitcoin wallet with a program encoding a check: “this wallet containing 1000BTC is only decryptable if you possess a proof of the Riemann Hypothesis."
- **Obfuscation** allows me to "encrypt" a program. This means scrambling the program such that someone else can run it on any inputs of their choosing and produce the correct outputs, but they can't learn anything[1] about the program’s internal state or structure.



A simplified “technology tree” of some cryptographic primitives. Terms in bold are programmable cryptography. Arrows represent reductions—in other words, primitives higher on the tech tree are more advanced, and generalize the nodes that point to them. Inspired by the Complexity Zoo.

The first instances of programmable cryptography have just started to become practical for developers in real-world use cases over the past five or so years[2]. In the coming decade, the performance, accessibility, and capabilities of programmable cryptography will increase dramatically.

The transition from special-purpose hardware to programmable, general-purpose hardware was a landmark shift for computing—think of the leap from an alarm clock to a CPU. As a result of these recent advancements in cryptography, we are on the cusp of a similar transition, but at the level of information rather than physical hardware.

What might the transition from special-purpose to programmable cryptography enable, and how will we get there?

What can we do with programmable cryptography?

When we think of cryptography today, we often think of words like *privacy*, *security*, or *integrity*. Special-purpose cryptography has historically been a tool for defending existing systems from bad actors.

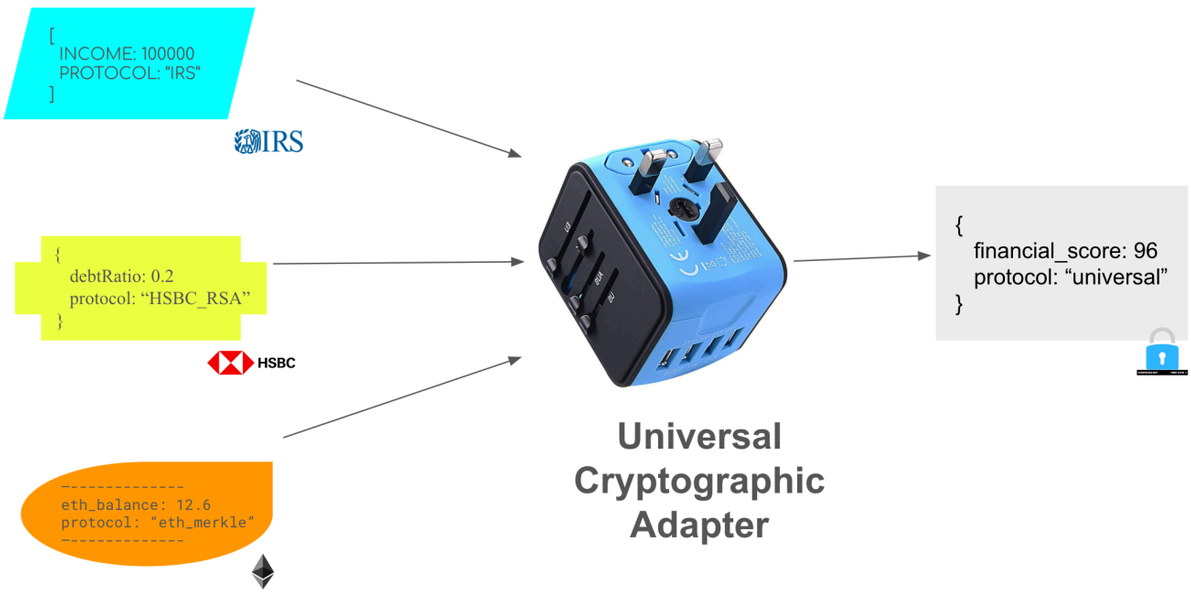
In contrast, due to its flexibility, general-purpose cryptography enables us to build computing systems with powerful *new* properties, that couldn't have existed before. Here are three examples.

The Universal Protocol

Imagine a single Universal Protocol for all of your social data, digital identity, financial interactions, professional history, medical data, and everything and anything else attached to you. The Universal Protocol specifies a universal data format, and a procedure for verifying the correctness and origin(s) of any data. The Universal Protocol is also extraordinarily flexible--in fact, it's Turing complete. Any computable function that is run on Universal Protocol data will result in an output that also conforms to the protocol, and that retains its cryptographic integrity.

Every website on the Internet handles and outputs data in a way that is compatible with the Universal Protocol. Additionally, any website or machine on the Internet can verify, understand, and consume data that conforms to the protocol[3]. You can combine Universal Protocol data packets together, transform them using any computable function, partially obfuscate them, hand them off to someone else to compose with their data, and in general manipulate them however you'd like; the resulting output will remain perfectly verifiable and interpretable to anyone else who also understands the Universal Protocol.

This might sound like a pipe dream, but with programmable cryptography this is theoretically within reach—without even requiring existing companies and service providers to switch over to new standards. In the past two years, we've developed tools from programmable cryptography to build "Universal Cryptographic Adapters,"[4] which in turn would enable us to transform any data to and from a Universal Protocol format. And as mentioned previously, the majority of the world's web traffic today already passes through special-purpose cryptographic protocols like HTTPS or DKIM. So any Internet user could "proxy" their HTTPS traffic through a universal adapter, and convert it into a format that is understood by any desired target platform or data consumer.



Universal Cryptographic Adapters, based on technologies like zkSNARKs, allow us to compose and transform data originating from any source cryptographic protocol/format to any destination.

Multiple very promising efforts in this direction, like TLSNotary and the Proof Carrying Data framework, are already approaching production today. These efforts are mostly based off of zkSNARKs, though more advanced gadgets like recursive ZK protocols and Multi-Party Computation will also be required to make even more general kinds of data transformations possible.

Of course, an enormous amount of engineering and infrastructure work would be required to realize the full vision of a Universal Protocol at scale. However, such transformations have happened before; if this pattern is recognized widely enough, one could imagine widespread adoption of a descendent of HTTPS (ZTTPS?) that's optimized to make this kind of universal composability even easier and more accessible.

Example applications of the Universal Protocol:

- Port your social data across all social platform--you can move your likes, friends, followers, history, and reputation seamlessly across Facebook, Twitter, Reddit, Instagram, etc.
- Financial intermediaries such as Equifax, Transunion, or Experian are replaced with cryptographic protocols on every consumer device; your smartphone can generate a cryptographically secure response to any query about your financial history or health, while keeping your underlying data private and in your own control.
- A network of universal and interoperable digital identity standards arise that are accepted by all websites, businesses, government agencies, universities, and more. These standards are permissionless--they rely on cryptography, rather than needing everyone to register with or query a specific authority.

Hallucinated Servers

Suppose that you and a handful of your friends, coworkers, or classmates want to set up a social networking site for your community. Your community wants to use this site to host public and private discussions, broadcast life updates to each other, manage membership in various interest sub-groups, maintain a community reputation system, and more.

Today, the story for this would look something like: you'd start by finding a provider like Amazon Web Services and renting a cloud server. Then, you'd write some code specifying the intended backend logic of the application, and you'd deploy this code to the server. Finally, to use the application, everyone would communicate with the server. You (and Amazon) would have special access to the cloud server, but you'd promise not to look too closely at the server's state, and you might set the application up such that no one is encouraged to put anything *too* sensitive on the site. If you wanted to build something that deals with more private or sensitive data, especially for people outside of your social circle, you'd need to build up lots of history, reputation, and trust.

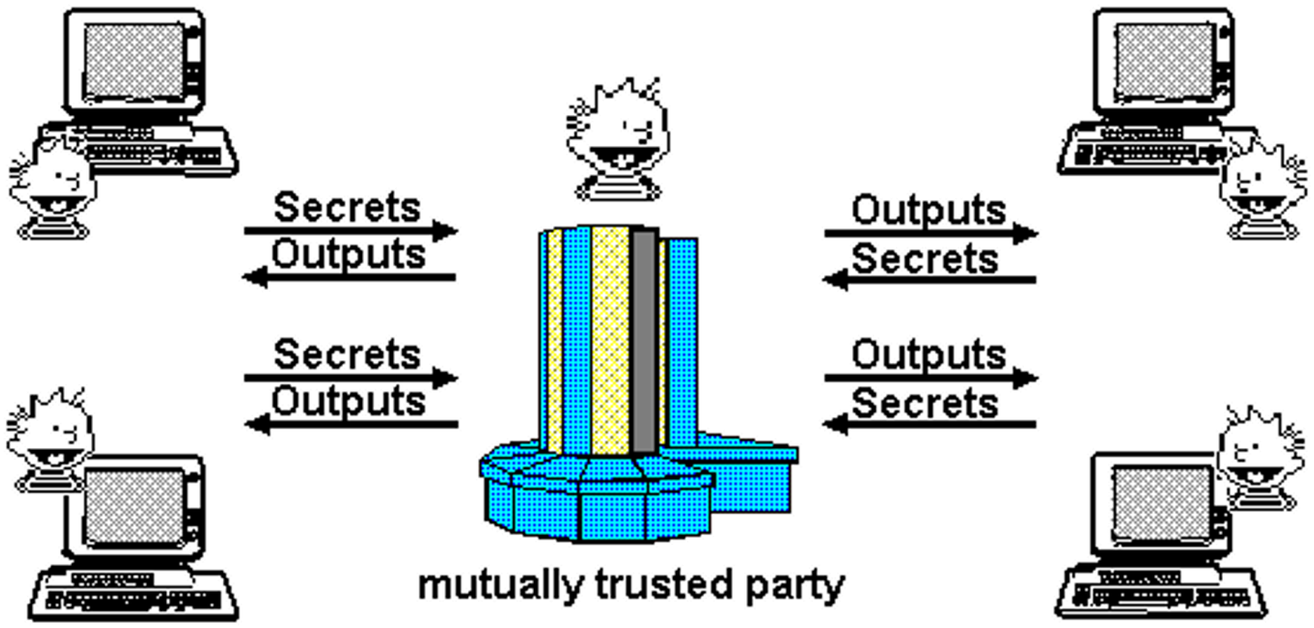
In a decade or two, the process of setting up and deploying applications on the Web might look very different. Many application backends might run inside VMs that are cryptographically "hallucinated" by their users at runtime.

Such digital services would have no physical footprint--there would be no particular physical machine where application state lives and code is executed. Instead, every user of an application would store their own private state alongside an encrypted "cryptographic shard" of the application's global state, and the userbase would collectively simulate the execution of programs on a "hallucinated server" by requiring that each participant execute some cryptographically obfuscated version of the program over their own shard. Programmable cryptography primitives can be used to ensure that users are able to read from, write to, and mutate state on this hallucinated server, but only according to pre-agreed upon rules encoded into server logic; a user wouldn't be able to read from someone else's private state, or from the "globally" private state held by the server.

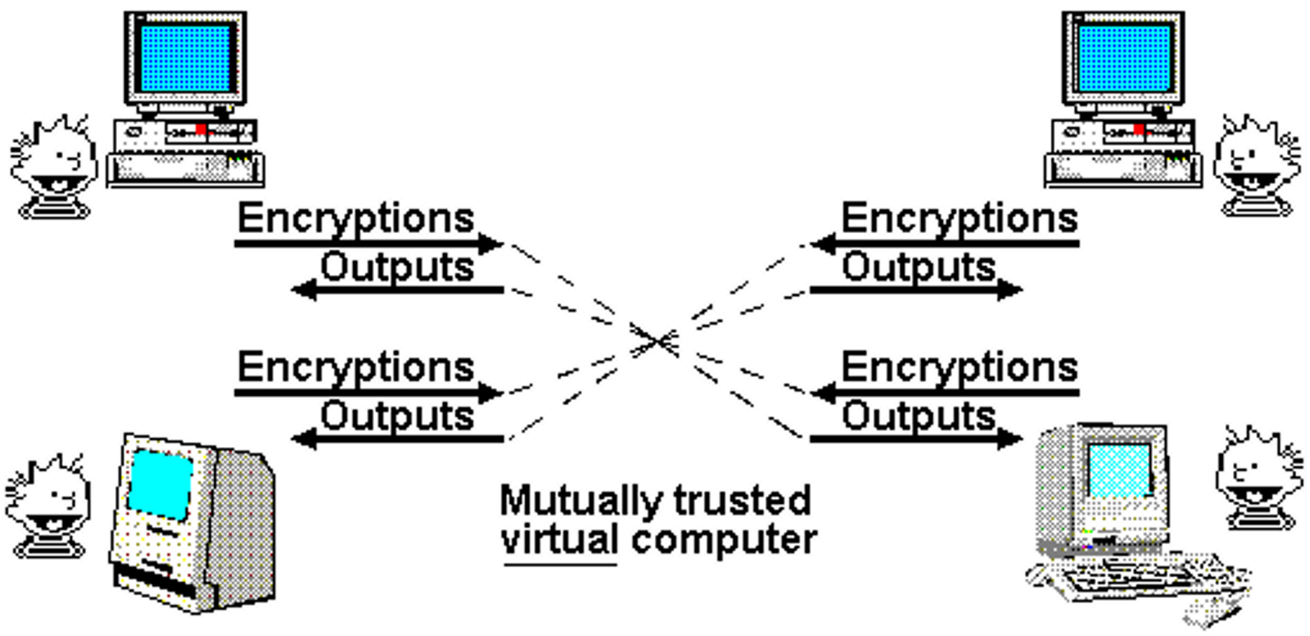
You could build services that execute over participants' data without ever interacting with the data directly; in fact, you could build virtual applications that maintain and compute over a state that *no one knows*!

Such a construction would clearly have far-reaching consequences for privacy, security, and user control of data. But the implications extend far beyond "merely" privacy and security. Today, our digital interactions are limited to a narrow, pre-specified set of allowed operations that a handful of centralized service providers have found viable business models and moats for. In this future, you could participate in whatever peer-to-peer or multi-peer interactions you'd like, by hallucinating the appropriate application backend with all other interested users at runtime. Networks of these hallucinated servers might make up a new cryptographic scaffolding for the

Internet: on top of this scaffolding, we could rapidly bootstrap new digital services that would today require a single service provider to build up brand capital, a locked-in userbase, a viable business model, and an enormous amount of trust over many years.



"Trusted Third Party" model. From Nick Szabo's 1997 essay, The God Protocols.



Contrast this with the "Mathematically Trustworthy Protocol" model, from the same essay.

This idea is a bit further out from the idea of the Universal Protocol / Universal Adapter, but only by a few years. Work on early prototypes of computing networks based on both general-purpose multi-party computation technology and fully homomorphic encryption is underway today. *Multi-party fully-homomorphic encryption* is a promising candidate for a base technology for these networks; further down the line, techniques like *program obfuscation* could decrease the communication complexity and liveness requirements for such networks.

Example applications of Hallucinated Servers:

- Discover personalized recommendations for a product, a movie, a job, or even a date with a personalized cryptographic agent that is constantly searching on

your behalf, without your data ever leaving your computer.
- Spin up a new, virtual social media platform "on the fly" for a community you're in, that hooks into all of your data from other social media platforms, without needing to run a server yourself *or* rely on any centralized intermediary to do so.
- Automatically get notified that you are at risk of some disease / medical outcome, without needing to share your medical data with anyone.

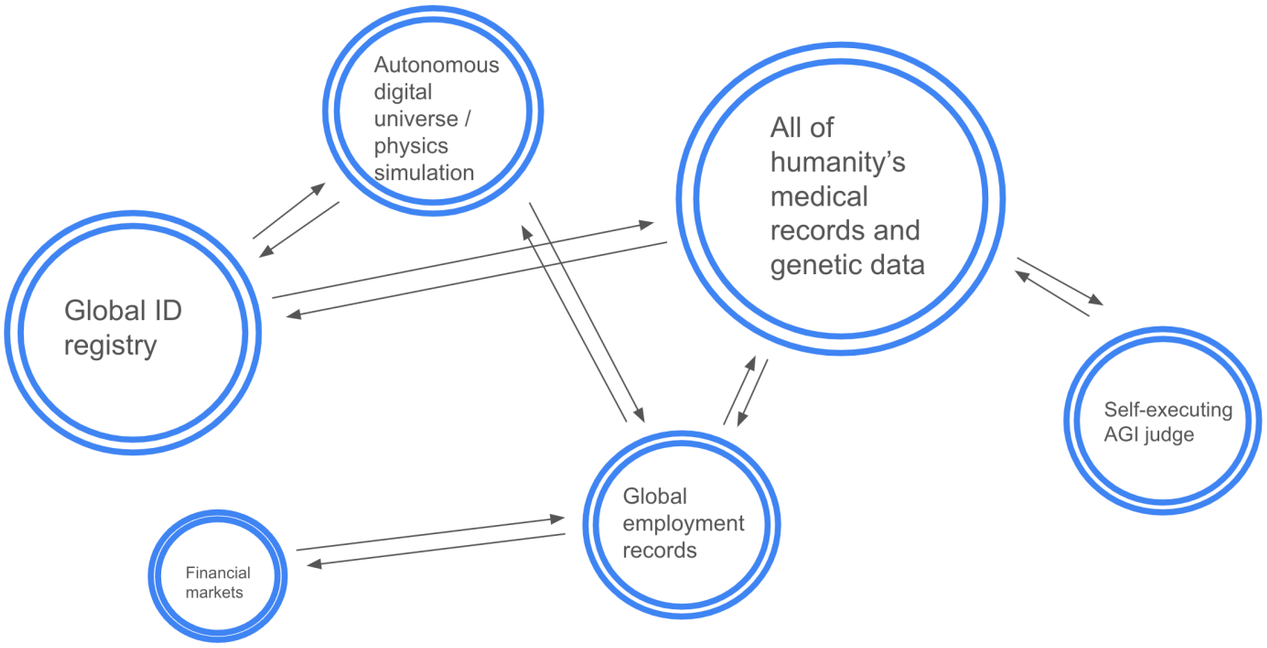
Cryptomata

The most powerful branches of programmable cryptography—like obfuscation, advanced primitives from quantum cryptography, and future descendents of modern blockchains—could allow us to build *cryptomata*.

Cryptomata (short for *cryptographic automata*) are autonomous, self-executing programs with perfect guarantees of correctness and privacy. No one can stop or tamper with the execution of a cryptomata; additionally, no one can “pry open” a cryptomata to read its internal state or memory. These guarantees are not just enforced by software, or even by hardware--they are enforced by math. They can never go down, they execute with cryptographically verifiable correctness, they never leak data, and they expose permissionless APIs that anyone can call.

The limit case of this technology is extraordinarily powerful, and at least a little scary. For example, running AGI inside of a cryptomata would give us a truly self-sovereign “brain” that thinks its own private thoughts, shielded from the world, and that can’t be turned off—effectively, an independent and intelligent agent made of pure information. Running a detailed physics simulation inside of a cryptomata would give us an autonomous digital reality--a digital world with an existence independent of our own, and perhaps with digital lifeforms just as real as our physical ones.[5]

More practically, within a few decades, civilization’s critical digital public infrastructure may be running inside a network of cryptomata: identity registries, credential systems, repositories of biometric or genetic data, banks, marketplaces, virtual worlds, and more. I could upload my educational credentials and verified professional history into a job-finding cryptomata, which would securely store them and then notify me when an employer uploads a compatible job. Another cryptomata might securely store a huge repository of voluntarily-shared genetic data, and continually run correlations and other pattern-finding yet perfectly privacy-preserving queries over the aggregate dataset at the request of researchers.



A network of cryptomata, supporting civilization's core digital infrastructure. Each cryptomata is perfectly secure, autonomous, privacy-preserving of its internal contents, and interoperable with all other cryptomata. The specific design of these systems is likely to determine whether they creates huge headaches or untold usefulness for civilization.

For those familiar with blockchains, a cryptomata might be implemented as an obfuscated program running inside of a smart contract--essentially, a smart contract that keeps its own private state. Off of blockchains, a cryptomata could be implemented with a combination of obfuscation primitives, zkSNARKs, multi-party computation, and perhaps quantum cryptography. However, there are still many open research questions that need to be resolved in order for us to even have a satisfactory theoretical construction of a cryptomata--for example, one key theoretical limitation today is that a cryptomata built with the technologies described above cannot prevent itself from being copied and forked infinitely, meaning that certain kinds of information hiding are not possible.

Example applications of Cryptomata:

- Digital services running civilization's core digital infrastructure--identity registries, credential systems, repositories of sensitive biological data, marketplaces, and more--that are totally autonomous, secure, and deeply interoperable by construction.
- Self-executing, self-sovereign agents made of code, that can hold their own secrets.
- Simulated digital universes which continue to operate even when they aren't being observed, with mathematical/cryptographic guarantees on the integrity of their physics.

Why Group These Technologies Together?

Imagine going back 50 years and approaching the components of the modern computing hardware stack in isolation—rather than thinking about an integrated "computing system."

- **Non-volatile memory**: The most "obvious" applications of NVM in isolation include portable storage: floppy disks and USB drives. A narrow application perspective on NVM might focus on how to make smaller and more portable USBs.
- **Fiber-optic cables**: Fiber-optic cables enable higher bandwidth phone lines and television—it's about getting more television channels to more people.
- **Liquid Crystal Displays (LCDs)**: On their own, LCDs enable things like "making thinner displays"—going from CRTs to flat-panel displays.

Individually, these technologies have useful but limited applications. However, when considered in the wider context of Computing, each of these technologies becomes a part of a far bigger story. Non-volatile memory enables us to build hard drives for computers—storage that persists when a machine is shut down, a basic feature underlying all of our assumptions about personal computing. Fiber-optics enable the modern Internet—90%+ of Internet traffic passes through undersea fiber-optic cables. Finally, LCDs are the base technology for PC displays, enabling modern monitors, GUIs, and nearly all modern computing interface technology.

With the benefit of hindsight, it's obvious that the most interesting stuff happens when you can see these pieces as components of a whole, integrated Computing stack. Each of these technologies' primary impacts come from being a part of a larger system, which is why it would be a miss to approach them in isolation.

The same may be true of cryptographic computing as well. Asking "what can we do with zkSNARKs" or "what can we do with FHE" in isolation, without a more holistic perspective on programmable cryptography, might be akin to asking "what can we do with LCDs" without thinking more broadly about computing or the Internet.

An Integrated Cryptographic Computing System

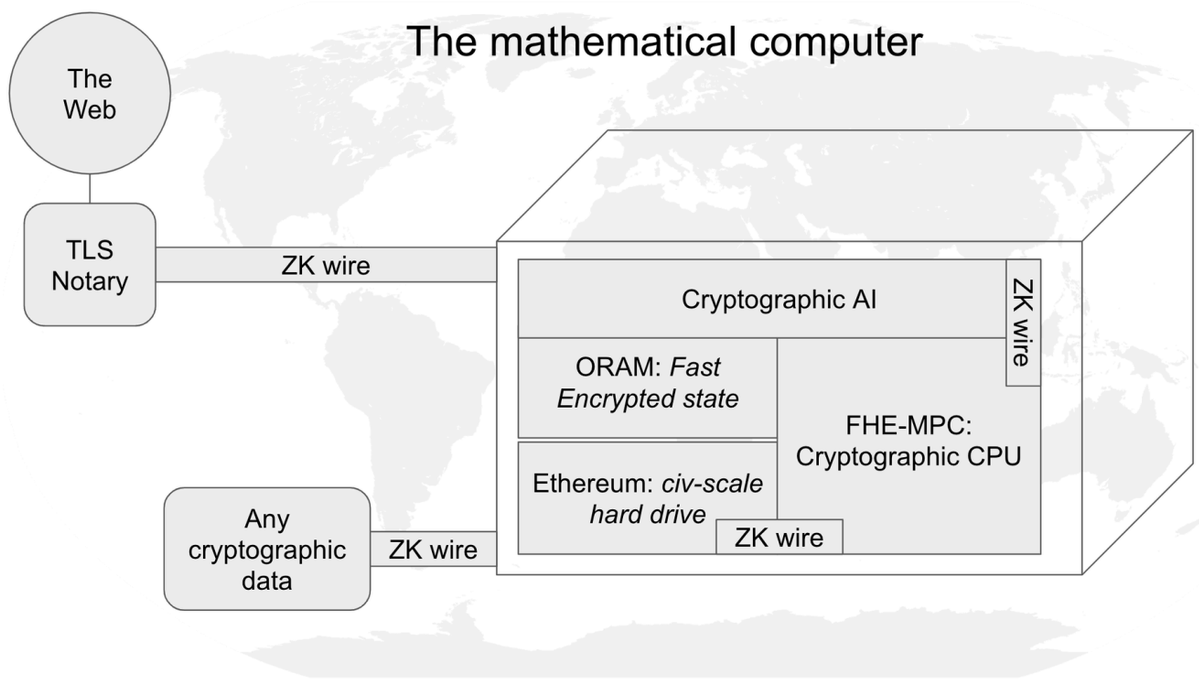


Diagram by Justin Glibert

In the past few years, people have started to build apps on top of programmable cryptography technologies. However, most of these applications have only involved one "branch" of programmable cryptography at a time. For example, products and companies are being built around the following ideas:

- **zkSNARKs** are useful for making blockchains faster (via systems like rollups and validiums).
- **Fully Homomorphic Encryption** (FHE) allows AI companies to train and perform inference over private, encrypted data.
- **Multi-Party Computation** (MPC) allows bidders on a marketplace (for example, farmers in a beet auction) to keep their bids private from sellers, or for hospitals to securely share sensitive data for research purposes.

These applications are definitely important. However, they are also quite narrow in various ways: in many of these cases, the cryptographic technologies are "merely" used to provide enhanced features on top of existing services. SNARKs are used to speed up blockchains, and FHE, MPC, and ORAM are used to add privacy or additional security on top of existing digital systems. This is a natural consequence of working backwards from a single primitive.

In contrast, the three ideas described in the previous section (the Universal Protocol, Hallucinated Servers, and Cryptomata) point to a world with fundamentally new computation and communication paradigms. **Notably, each of these systems also requires several (if not all) programmable cryptography primitives to work together, in order to be fully realized.**

For example, realizing the vision of Hallucinated Servers would require us to have verifiable encrypted computation—SNARKs on top of FHE and MPC. Inputs into a cryptographic negotiation in this world would ingest Universal Protocol data, which

relies on ZKPs. And in order to execute such operations statefully and performantly, we would likely need to integrate ORAM or similar technologies.

As this example shows, different programmable cryptography primitives offer different capabilities. SNARKs allow us to perform computations that are verifiably correct; FHE and MPC provide the ability to perform computation over confidential data; obfuscation and more exotic programmable encryption primitives allow us to reduce liveness requirements in networks. We need to understand what these primitives do and how they compose with each other in order to understand what kinds of systems we can build.

The speculative systems described in the previous section only scratch the surface of how we might think about programmable cryptography. We need a cohesive effort to understand the joint implications of various programmable cryptography tools, because the most impactful applications will require deep integration of multiple (and possibly all) of these primitives. Technologists, engineers, and researchers would benefit from seeing the individual branches of programmable cryptography as components of a unified, full-stack story.

Conclusion

It's been nearly 50 years since the invention of modern cryptography. Since then, we've gone from a world where no digital data is cryptographic, to a world where effectively all data on the Internet is cryptographic: all major websites use HTTPS; nearly all emails are signed with DKIM (required by Google as of February 2024!); an increasing proportion of our messages and cloud data is encrypted; over two trillion dollars of economic value is held in cryptographic tokens; and the key distribution problem is finally being solved at scale as well. First-generation cryptographic technologies have enabled a secure communications system that supports planet-scale information retrieval, messaging, commerce, and more.

Over the next 50 years, programmable cryptography may enable us to make the leap from cryptographic data to cryptographic *compute*: humanity's most important digital *programs*, and not just its digital traffic, may become cryptographic-by-default as well. Such a transformation would enable us to build a richer, more structured, and more democratic cyberspace. This upgraded cyber-medium would have many benefits: from greater privacy and control of data by users, to deep interoperability and frictionless communication between all digital services, to the power to create entire new digital universes.[6]

In future posts, we'll discuss the programmable cryptography "technology tree," how it might be advanced, and its relation to other technologies like blockchains.

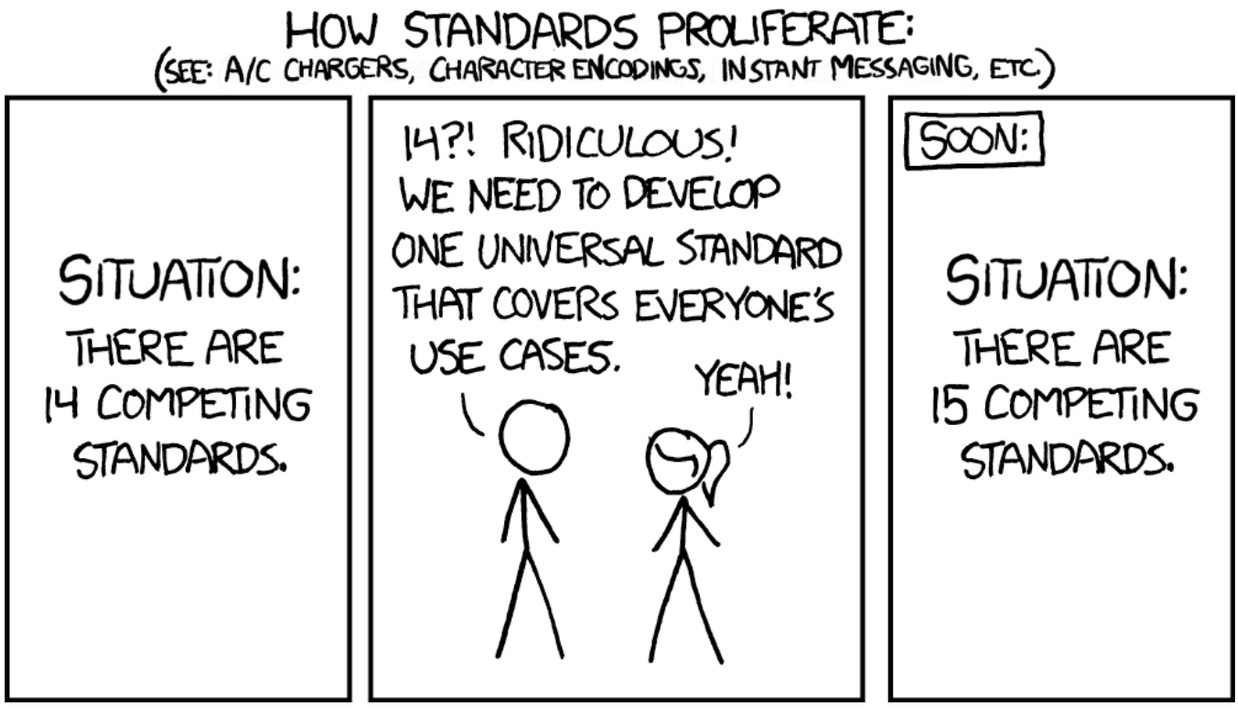
Footnotes

1. There are technical details about the theoretical limits of obfuscation; it's not quite accurate to say that you can hide anything. See this informal explanation. ↩

2. Different programmable cryptography primitives are at different stages of development. zkSNARKs, MPC, and FHE are in (or close to) production, while Witness Encryption and Obfuscation are still very theoretical, and current constructions are far too inefficient for practical use. ↩

3. The idea of the Universal Protocol has similarities to (and is partially inspired by) efforts like the Semantic Web. More broadly, Programmable Cryptography unlocks many architectures envisioned in the early days of the Web; we'll discuss this more in future posts. ↩

4. It may be more accurate to characterize this gadget as a "universal cryptographic adapter," or a "meta-protocol,"" instead of "universal protocol." In practice, it's likely that many different protocols and sub-protocols will arise: for example, different governments will probably issue digital ID credentials in different ways. The key idea is that it's now possible to compose and translate between them, by using programmable cryptography as an adapter. For example, you could use the adapter to compose some digitally signed data from the IRS, signed data from your bank, and a state proof from the Ethereum blockchain, into a single financial credential that can be verified and interpreted by an arbitrary third party in whatever format they'd like.



https://xkcd.com/927/ ↩

5. The idea of autonomously-executing digital realities is being explored by developers and creatives in the Autonomous Worlds community. Today, many people are building on-chain games as early explorations of simple Autonomous Worlds. You can read more about autonomous worlds here. ↩

6. Josh Stark has a useful lens on what exactly we get from tools like cryptography (and blockchain)--he calls this property *hardness*. See his blog post and talk on