# Smarter Signatures

Experiments in Verification

Christopher Allen, Shannon Appelcline
Berkeley, USA
ChristopherA@blockstream.com, ShannonA@skotos.net

***Abstract*—Digital signatures are a fundamental aspect of identity on the internet, used in technologies such as certificates, secure logins, and blockchains. However, today's signatures are simplistic: they can be improved to create more powerful and more complex signatures that are smarter.**

***Keywords—smart signatures; multisignatures; delegation; identity; predicate language; secure language***

## I. Introduction

The traditional usage of digital signatures is quite straightforward. The owner of a cryptographic identity signs a message (or a certificate) with his private key; a recipient can then use the related public key to verify the message.

However, digital signatures can do much more. Some technologies already support multisignatures, which can be signed by multiple people from a larger group. But even that doesn't support the full richness of business and computer logic that is becoming a part of our digital life. Simple signatures can't offer the flexibility that is needed by modern enterprises, and they can't offer the reliability that is required for modern finances.

To support these needs requires a new kind of signature — a *smarter signature* that increases options while still meeting the responsibilities of a robust and trusted signature system.

## II. Designing Smart Signature Systems

Designing a smarter signature system requires an understanding of the potential uses of smart signatures and the potential pitfalls.

### A. The Use of Smart Signatures

The core use of a signature is *verification*: a signature must ensure that the authorization conditions required for a task are met. In the world of simple signatures, that meant verifying that the right person signed a message. However smart signatures have a wider scope, supporting more use cases.

*1) **Multifactor Expressions.** A smart signature should support the inclusion of multiple elements within a single signature. This should include a variety of multisignatures — including N of N signatures, M of N signatures, and signatures with logical ORs. It should also include other varied signature elements, such as biometric signatures and proof of hardware control.*

*2) **Signature Delegation.** A key holder should be able to precisely control how a smart signature is used. He should be able to delegate its use to others with limits based on time, use, or content, and he should be able to pass on control of a smart signature if his own usage of a key ceases.*

*3) **Internal Depth.** A smart signature should support internal depth by combining these different possibilities. For example, a development release of software could include both multifactoring and delegation by requiring 3-of-5 signatures, where one signer has authorized his assistant because he's on leave, and another signer requires 2-of-2 keys for his signature, one of which is stored on a hardware token. Because this depth is created through internal links, the requirements are all evaluated synchronously.*

*4) **Transactional Support**. A smart signature system should support external depth by being able to prove that specific states have been reached in a larger state machine through the chaining of multiple signatures. For example, an art dealer might need to examine the transactional history of a painting to ensure that he's not purchasing stolen goods. Because this depth is created through external links, the requirements tend to be evaluated asynchronously: one smart signature at a time.*

These use cases all focus on the *creation* of signatures, providing functionality that signers need. However, there are always two users for any signature: the signer and the verifier. Additional verifier-focused use cases may illuminate UI and UX requirements for a smart signature system.

### B. The Requirements of Smart Signatures

Because smart signatures offer increased complexity over simple signatures, care must be taken to avoid security pitfalls. To ensure this, six requirements are suggested for the creation of smart signature systems:

*1) **Composable.** The increased complexity of smart signatures requires that they be built using some sort of programming language. However, the language itself must remain simple, with complexity built up from a constrained set of operations*

*2) **Inspectable.** Signatures must be easily understandable by a qualified programmer, so that any sophisticated user can readily evaluate the elements of a signature and how they will be verified. This requirement often emerges naturally from composability.*

*3) **Provable.** Signatures must be formally analyzable, so that they can support logical reasoning and so that sophisticated users and expert computer tools can have foreknowledge of the requirements of verification.*

*4) **Deterministic.** Signatures must always produce the same results, even when run on different machines or different operating systems.*

*5) **Bounded.** Signatures must not be able to exceed appropriate CPU or memory limitations through creation of malicious (or bad) signings. They need to minimize their size in order to minimize bandwidth and storage costs. Additionally, enforcement of these limitations must be deterministic.*

*6) **Efficient.** Though we place no requirements on the difficulty of creating signatures, the cost of verifying them should be very low.*

Together, these requirements insure the security of the signature language, of the individual signatures, and of the system running the signatures.

The element of ***privacy*** should also be considered. In smart signature design, there is a trade-off between flexibility and fungibility: many of the functions that make signatures smarter also require participants to reveal more about who they are, reducing the substitutability of the persons involved in the signatures and of any resources being signed. Even if privacy is not a *requirement*, it should be a *consideration*; any decisions about the level of privacy in a signature system should be known and purposeful.

## III. Experiments in Smart Signatures

Fulfilling these uses and meeting these requirements for smart signatures necessitates the creation of better languages and better tools. A number of promising program languages each have their own strengths and weaknesses.

### *A. Bitcoin Script*

Bitcoin Script already exists and is being used to safeguard millions of dollars worth of transactions [1]. It currently authorizes the spending of bitcoins, primarily with single signatures or M-of-N multisignatures. However, it's possible to encode more complex redemption conditions into a Bitcoin Script, and even to keep them secret — allowing a recipient to prove that he met the signing conditions by matching a hash of those conditions. Though currently used on the blockchain, Bitcoin Script could be used as a the basis of a generalized smart signature language outside of those contraints.

### *B. Dex*

Peter Todd is working on another possible system for smart signatures, one that he calls Dex, a system of deterministic predicate expressions [2] [3]. As the name suggests, it's deterministic, guaranteed to always return the same result for a specific signature and environment. Dex also more fully embraces functional programming thanks to its basis in lambda calculus. Functional programming languages tend to be an excellent choice for smart signature languages: besides being deterministic, they're also composable and provable.

### *C. Crypto Conditions*

Crypto-conditions [4] were developed by Stefan Thomas as part of the Interledger project. The protocol relies on one or more ledgers that are involved in an end-to-end transfer being able to put funds on hold pending the fulfillment of a predefined condition. This condition is, in effect, the definition of a smart signature, and the fulfillment of that condition is the signature itself.

### *D. Sequent Calculus*

Sequent Calculus [5] [6] offers another approach to smart signatures. It can create smart signatures as formal proofs, where simpler proofs are functionally combined to ultimately create smarter signatures that are analyzable formally. This methodology is currently being studied by another author, with a full paper pending.

## IV. Conclusion

Whatever language is used, a smart signature system that supports these uses, that meets these requirements, and that considers its privacy implication, would add powerful tools to the digital world by meeting the needs of the financial and business worlds.

## Acknowledgment

This paper is based on the longer "Smarter Signatures: Experiments in Verifications" [7] that was an output of the Rebooting the Web of Trust II: ID2020 Design Workshop.

## References

[1] C. Allen, G. Maxwell, P. Todd, R. Shea, P. Wuille, J. Bonneau, J. Poon, and T. Close. "Smart Signatures". Rebooting the Web of Trust I. *https://github.com/WebOfTrustInfo/rebooting-the-web-of-trust/blob/master/final-documents/smart-signatures.pdf*. 2015.

[2] P. Todd. "Dex: Deterministic Predicate Expressions for Smarter Signatures". Rebooting the Web of Trust II: ID2020 Workshop. *https://github.com/WebOfTrustInfo/ID2020DesignWorkshop/blob/master/topics-and-advance-readings/DexPredicatesForSmarterSigs.md*. 2016.

[3] P. Todd. "Building Blocks of the State Machine Approach to Consensus". Peter Todd. *https://petertodd.org/2016/state-machine-consensus-building-blocks*. 2016.

[4] S. Thomas. "Crypto Conditions". GitHub. *https://github.com/interledger/rfcs/tree/master/0002-crypto-conditions*.

[5] Ariola, Zenn M., Aaron Bohannon, and Amr Sabry. 2009. "Sequent Calculi and Abstract Machines". ACM Transactions on Programming Languages and Systems. *http://www.cs.indiana.edu/~sabry/papers/sequent.pdf*.

[6] N. Guenot and D. Gustafsson. "Sequent Calculus and Equational Programming". IT University of Copenhagen. *http://arxiv.org/pdf/1507.08056.pdf*. 2015.

[7] C. Allen, S. Appelcline. "Smarter Signatures: Experiments in Verification". Rebooting the Web of Trust II. *https://github.com/WebOfTrustInfo/ID2020DesignWorkshop/blob/master/final-documents/smarter-signatures.pdf.* 2016.