Quantum Tokens for Digital Signatures

Shalev Ben-David,[1] Or Sattath[2,3]

[1]Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology
[2]The Benin School of Computer Science & Engineering, The Hebrew University
[3]The Center for Theoretical Physics, Massachusetts Institute of Technology

Abstract

The fisherman caught a quantum fish. *Fisherman, please let me go*, begged the fish, *and I will grant you three wishes.* The fisherman agreed. The fish gave the fisherman a quantum computer, three quantum signing tokens and his classical public key. The fish explained: *to sign your three wishes, use the tokenized signature scheme on this quantum computer, then show your valid signature to the king who owes me a favor.*

The fisherman used one of the signing tokens to sign the document "give me a castle!" and rushed to the palace. The king executed the classical verification algorithm using the fish's public key, and since it was valid, the king complied.

The fisherman's wife wanted to sign ten wishes using their two remaining signing tokens. The fisherman did not want to cheat, and secretly sailed to meet the fish. *Fish, my wife wants to sign ten more wishes.* But the fish was not worried: *I have learned quantum cryptography following the previous story*. These quantum tokens are consumed during the signing. Your polynomial wife cannot even sign four wishes using the three signing tokens I gave you.*

How does it work? wondered the fisherman. *Have you heard of quantum money? These are quantum states which can be easily verified but are hard to copy. This tokenized quantum signature scheme extends Aaronson and Christiano's quantum money scheme, which is why the signing tokens cannot be copied.*

Does your scheme have additional fancy properties? asked the fisherman. *Yes, the scheme has other security guarantees: revocability, testability and everlasting security. Furthermore, if you're at sea and your quantum phone has only classical reception, you can use this scheme to transfer the value of the quantum money to shore*, said the fish, and swam away.

*The Fisherman and His Wife by the brothers Grimm.

The full version of this QCrypt extended abstract is available on `https://arxiv.org/abs/1609.09047`.

Introduction One of the main goals of cryptography is to allow an authorized party, typically holding a secret key, to perform an action which an unauthorized party (without the key) cannot. For example, in a digital signature scheme, the authorized party, Alice, holds a secret key that allows her to create digital signatures that will be accepted by a public verification algorithm. Anyone without Alice's key cannot forge her signature.

In this work, we consider the task of delegating *limited* authorization: is it possible to provide a one-time access to the secret key to a third party? For example, if Alice goes on vacation, can she allow Bob to sign one (and only one) document of his choice?

Classically, Bob either knows the secret key or doesn't, and there is no way to control how many times the key is used. But with quantum mechanics, the situation is different: the no cloning theorem [1] allows us to create secrets that cannot be copied. Consequently, we propose the design of cryptographic schemes with two levels of secrets: one classical "master" secret, which is used only to generate any number of unclonable quantum "tokens", each of which can be used to perform one action, and is consumed in the process due to destructive effects of quantum measurements. If Alice holds the secret key, she can delegate authorization to Bob by granting him a limited number of quantum tokens.

Tokens for Digital Signatures This work applies the previous proposal specifically to digital signatures, allowing the delegation of limited authorization via the use of quantum tokens.

Digital signature is a cryptographic primitive which is arguably second in importance only to encryption. A digital signature scheme [2] consists of three Probabilistic Polynomial Time (PPT) algorithms: $\mathsf{key\text{-}gen}$, sign, and verify. The first algorithm outputs a secret key sk and a public key pk. The signer can use sk to sign a document α by calling $\mathsf{sign}(sk, \alpha)$. This produces a signature sig, which can be verified by anyone holding the public key by calling $\mathsf{verify}(pk, \alpha, sig)$. We will denote the keys as subscripts, so these calls are $\mathsf{sign}_{sk}(\alpha)$ and $\mathsf{verify}_{pk}(\alpha, sig)$. The verify algorithm returns either "accept" or "reject". A valid signature should be accepted, so $\mathsf{verify}_{pk}(\alpha, \mathsf{sign}_{sk}(\alpha))$ should accept for all α. Informally, a digital signature scheme is *secure* if an adversary which can ask for signed documents, cannot efficiently forge any *new* signed document.

Our main contribution is a construction of a *tokenized* signature scheme. A tokenized signature scheme consists of 4 quantum polynomial time (QPT for short) algorithms: $\mathsf{key\text{-}gen}$, $\mathsf{token\text{-}gen}$, sign, and verify. In this setting there are *three* entities: a signer, a verifier and a new entity, which we will call the signing authority. In this context, even though the signer is the one signing the document, it is done in the name of the signing authority. The authority generates the pair (sk, pk) using $\mathsf{key\text{-}gen}$ as before. Next, it generates a quantum state $|♙\rangle$, which we call a *signing token*, by running $\mathsf{token\text{-}gen}_{sk}$. Note that different calls to $\mathsf{token\text{-}gen}_{sk}$ may produce different signing tokens. The signer, who gets one copy of a signing token from the authority, can sign a single document of her choice. The output of $\mathsf{sign}(\alpha, |♙\rangle)$ is a bit-string, as in the classical setting. The correctness property remains unchanged: $\mathsf{verify}_{pk}(\alpha, \mathsf{sign}(\alpha, |♙\rangle))$ must accept for all documents α.

The novelty of tokenized signatures is that sign applies a measurement which collapses $|♙\rangle$, and therefore it cannot be reused to sign an additional document. Informally, the security requirement is that a QPT adversary Adv, with access to the public key pk and to ℓ signing tokens $|♙_1\rangle \otimes \ldots \otimes |♙_\ell\rangle$, cannot generate valid signatures for $\ell + 1$ different documents.

Here are two motivating examples. (i) A manager wants to hedge the embezzlement risks that an accountant introduces. This can be achieved by agreeing with the bank that any signed wire is limited to \$1000. Then, the manager can grant the accountant a number of tokens according to the level of her trust in the accountant. (ii) Online computers are more prone to hacks than offline computers. A system administrator can hold the secret keys on an offline computer, and generate signing tokens to be used on the online computer. The hacker who steals n signing tokens can sign at most n documents, whereas a stolen secret key can be used to sign any number of documents.

Tokenized signatures have several other useful properties. A Tokenized signature scheme can be used as a digital signature scheme. Every tokenized digital scheme is also *revocable*: the signer can destroy a token in a publicly-verifiable way. Our construction has an even stronger notion of revocability, testability, which allows testing the signing token without consuming it.

If the adversary has no access to the public key, revoke and $\mathsf{verify\text{-}token}$ in some of our schemes have *everlasting security*, which means that a computationally unbounded quantum adversary with access to a single copy of $|♙\rangle$ but without access to pk, cannot pass revocation, while also generating a valid signature for some document α.

A Candidate Construction It turns out that the main challenge is to construct a weak scheme, which supports signing only 1-bit documents, and that is secure only against an adversary with access to a single signing token. We strengthen any such weak scheme to a full tokenized signature scheme (which is secure against adversaries with many tokens, and each token can be used to sign an arbitrary long document) using standard techniques.

Our weak scheme is based on Aaronson and Christiano's quantum money scheme [3]. Roughly speaking, the construction of Aaronson and Christiano works as follows. They consider a random subspace A of $\mathbb{F}_2^n$ of dimension $n/2$, and the money state (in our case, this would be the signing token) uses n qubits, which are in the uniform superposition over A, denoted by $|A\rangle$.

Another important subspace of $\mathbb{F}_2^n$ is $A^\perp$:

$$A^\perp = \{b \in \mathbb{F}_2^n | \forall a \in A,\ a \cdot b = \sum_{i=1}^{n} a_i b_i \mod 2 = 0\}.$$

Applying $H^{\otimes} n$ on $|A\rangle$ gives the state $|A^\perp\rangle$:

$$H^{\otimes n}|A\rangle = |A^\perp\rangle = \frac{1}{2^{n/4}} \sum_{b \in A^\perp} |b\rangle.$$

We can use the above properties to construct a *private* tokenized signature scheme. In the classical literature, there are two types of digital signatures: the standard one, which was described above; the other is a private (symmetric) digital signature scheme, known as *message authentication code* (MAC), in which verification requires the secret key.

The signing authority samples a random $n/2$ dimensional subspace A as the secret key, and generates $|A\rangle$ as the signing token. The idea is to let any non-zero element in A correspond to a signature for the bit 0, and a non-zero element in $A^{\perp}$ correspond to a signature for the bit 1. An adversary holding $|A\rangle$ can either measure it in the standard basis to get a (uniformly random) element of A, or can measure $|A^{\perp}\rangle$ to get an element of $A^{\perp}$. We show that an adversary with a single copy of $|A\rangle$ cannot do both, since measuring the state collapses it. This is a manifestation of the no-cloning theorem [1]: if the adversary could clone the state $|A\rangle$, and hold two copies of it, he could use the first copy to find an element of A, and the second copy to find an element of $A^{\perp}$, and break the security of the scheme. The verifier, which in the private setting knows the secret key (the choice of A and $A^{\perp}$), can verify whether the signature is valid, by testing for the relevant subspace membership.

How can we turn this into a public tokenized signature scheme? Suppose the adversary has one copy of $|A\rangle$ as before, but additionally can ask questions of the form: is x in A? Is y in $A^{\perp}$? We use quantum query complexity techniques to prove that such an adversary still cannot efficiently find both a non-zero element of A and of $A^{\perp}$. These questions are sufficient to verify signatures. Therefore, the problem we face is the following: is there a way to obfuscate the membership for A and $A^{\perp}$? The obfuscated program should allow the verifier to test whether an element is in the relevant subspace, but should not leak any additional information about A and $A^{\perp}$. Aaronson and Christiano faced the same challenge and suggested an ad-hoc approach for obfuscating these subspaces. Unfortunately, their construction is broken (see [4] and the main text), and therefore cannot be used.

We define a version of virtual black-box obfuscation for subspaces which would be sufficient to prove our scheme secure, and argue that the existence of such an obfuscation scheme is plausible (for instance, there are no known impossibility results that apply). Unfortunately, there are no candidate constructions which satisfy this requirement. We then conjecture that the indistinguishability obfuscation ($i\mathcal{O}$), for which there are candidate constructions, satisfies this definition; under this conjecture, we can prove our scheme secure.

Computational requirements One of the functions of money is store of value. From an engineering perspective, this makes quantum money (which our construction is based on) a challenging task, as it requires a long term quantum memory to store the quantum money. Signing tokens do not necessarily need to be stored for long periods (perhaps, only the signed documents are), and therefore some applications may require relatively short term memory, and will thus be easier to implement in practice. Furthermore, signing requires a very simple (depth 1) quantum circuit, and verification is done on a classical computer. Our tokens can be generated using Clifford circuits, which do not require a universal quantum computer, and therefore may be easier to implement. The only task which requires a universal quantum computer is `verify-token`. The security of the weak scheme is exponential in the number of qubits; this means, among other things, that the depth of `token-gen` in all our schemes is only super-logarithmic (in the security parameter).

Application to Quantum Money We show that a testable tokenized signature scheme can be used as quantum money[5, 6, 7, 8, 9, 3, 10, 11, 12]. Interestingly, the quantum money scheme we get this way has a combination of desirable properties that no previous scheme had, not even the Aaronson-Christiano scheme on which our construction is based.

To get a quantum money scheme from a testable tokenized signature scheme, simply use the signing tokens as money. An interesting property of this scheme is that the money can be converted into a verifiable signature for a document. This can be used to send the money over a classical channel (vis-à-vis "standard" quantum money, which can only be sent via a quantum channel).

In particular, suppose Alice wants to send money to Bob, but she is stranded and cannot get quantum internet reception on her quantum cellphone. With our scheme, Alice can use her money as a signing token to sign the document "I'm sending this money, with the serial number 03217, to Bob". Bob can then take the signed document and present it to his bank. The bank knows that Alice must have burned her money to produce the signed document, and so it can safely issue Bob a new money state. In essence, Alice can convert a quantum coin into a classical check. We discuss other advantages of this schemes related to fraud and other attack vectors.

References

[1] Wootters, W. K. & Zurek, W. H. A single quantum cannot be cloned. *Nature* **299**, 802–803 (1982).

[2] Diffie, W. & Hellman, M. E. New directions in cryptography. *IEEE Trans. Information Theory* **22**, 644–654 (1976).

[3] Aaronson, S. & Christiano, P. Quantum money from hidden subspaces. *Theory of Computing* **9**, 349–401 (2013).

[4] Pena, M. C., Faugère, J. & Perret, L. Algebraic cryptanalysis of a quantum money scheme the noise-free case. In Katz, J. (ed.) *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, vol. 9020 of *Lecture Notes in Computer Science*, 194–213 (Springer, 2015).

[5] Wiesner, S. Conjugate coding. *ACM Sigact News* **15**, 78–88 (1983).

[6] Bennett, C. H., Brassard, G., Breidbart, S. & Wiesner, S. Quantum cryptography, or unforgeable subway tokens. In *Advances in Cryptology*, 267–275 (Springer, 1983).

[7] Tokunaga, Y., Okamoto, T. & Imoto, N. Anonymous quantum cash. In *ERATO Conference on Quantum Information Science* (2003).

[8] Mosca, M. & Stebila, D. *Quantum coins*, vol. 523 of *Contemp. Math.*, 35–47 (Amer. Math. Soc., 2010).

[9] Aaronson, S. Quantum copy-protection and quantum money. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC 2009, Paris, France, 15-18 July 2009*, 229–242 (2009).

[10] Gavinsky, D. Quantum money with classical verification. In *IEEE 27th Annual Conference on Computational Complexity*, 42–52 (IEEE, 2012).

[11] Farhi, E., Gosset, D., Hassidim, A., Lutomirski, A. & Shor, P. Quantum money from knots. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, 276–289 (ACM, 2012).

[12] Georgiou, M. & Kerenidis, I. New constructions for quantum money. In Beigi, S. & König, R. (eds.) *10th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2015, May 20-22, 2015, Brussels, Belgium*, vol. 44 of *LIPIcs*, 92–110 (Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015).