QUANTUM TOKENS FOR DIGITAL SIGNATURES

By Shalev Ben-David and Or Sattath (2017)

Elad Prager
Daniel Kovachev
Omer segal

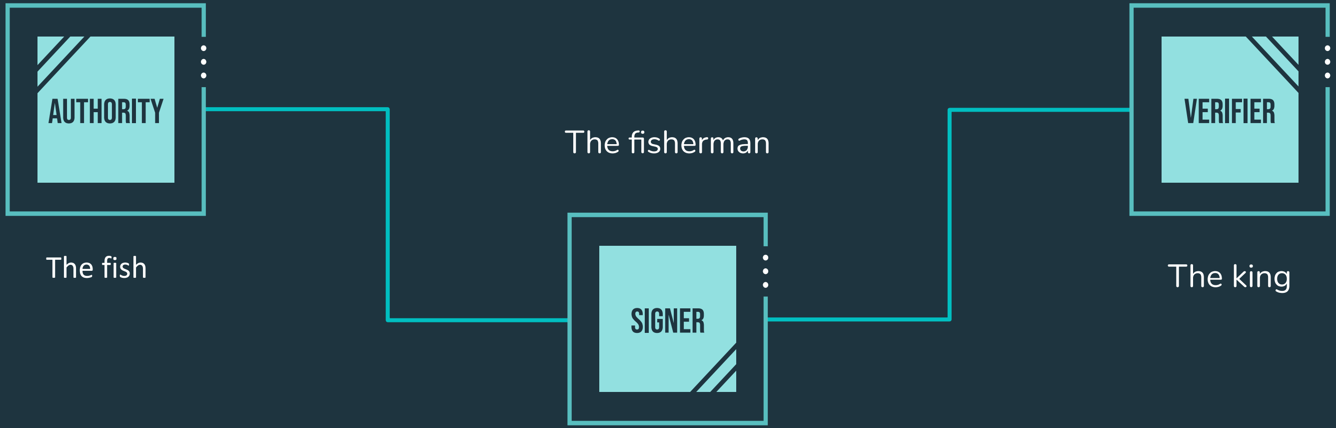
Quantum Cryptography Course

Motivation

1. A fish gives a fisherman 3 **wishes** so that the fisherman will spare his life.
2. The fisherman can now use these wishes to **ask anything from the king**.
3. The king can **verify** that the fish indeed gave the authority.
4. The fisherman **cannot produce another wish** from the given wishes.

The Goal: Delegate **another party** to sign a **limited** number of documents.

Quantum Tokens



Digital signature (classical Signature)

Syntax

KEYGEN

Generate a secret and a public key (given security parameter κ)

sk and pk

SIGN

Sign a document α with sk

sig

VERIFY

Verify (α, sig) with pk

accept/reject

Digital signature (classical Signature)

Correctness

$$Pr\left[verify_{pk}\left(sign_{sk}(\alpha)\right)=T\right]=1$$

Security

$$Pr\left[(\alpha,s)\leftarrow Adv^{Sign_{sk}}(pk),\ verify_{pk}(\alpha,s)=T\wedge\alpha\in Q_{Adv}^{Sign_{sk}}\right]\leq negl(\kappa)$$

- Where Q is the set of queries an adversary made to the oracle and **Adv is a QPT** algorithm with access to pk

Tokenized signature scheme

Syntax

KEYGEN

Generate a secret and a public key(given security parameter κ)

↓

sk and pk

TOKEN-GEN

Generate a signing token(quantum state) using sk

↓

τ

SIGN

Sign a document α with τ

↓

sig

VERIFY

Verify (α, sig) with pk

↓

accept/reject

Tokenized signature scheme

Correctness

$$Pr\left[\tau \leftarrow token - gen_{sk}, verify_{pk}(\alpha, sign(\alpha, \tau)) = T\right] \geq 1 - negl(k)$$

Security

- $\left(m_1, \sigma_1, \ldots, m_{l+1}, \sigma_{l+1}\right) \leftarrow QAdv\left(pk, |\tau_1\rangle \otimes \ldots \otimes |\tau_l\rangle\right)$
- $Pr\left[verify_{l+1,pk}\left(m_1, \sigma_1, \ldots, m_{l+1}, \sigma_{l+1}\right) = T \wedge \forall i \neq j : m_i \neq m_j\right] \leq negl(k)$
- **one time** tokenized signature scheme: l = 1

Oracle

- An oracle is a ‘function’ you query in a **black box** manner.
- The following scheme is **secure relative to an oracle.**
- This construction **cannot be instantiated**.

Construction Definitions

- 1 bit scheme: sign a document of size 1 bit.
- Restricted scheme: sign a predetermined size document.
- Unrestricted scheme: sign a document of any size.

Construction Overview

Intuition: reduce the general problem to one time 1 bit scheme and extend gradually.

- One Time tokenized scheme for 1 bit
- One Time tokenized scheme for restricted number of bits from 1 bit scheme
- One Time tokenized scheme for unrestricted number of bits from restricted scheme
- Tokenized Signature Scheme from unrestricted scheme.

One time 1 bit Scheme - Intuition

- We will use Aaronson and Christiano **hidden subspace scheme**.
  - Note: Any secure construction will qualify as a candidate for the 1 bit scheme.
- We will later show that this **construction is secure relative to an oracle**.

One time 1 bit Scheme

Aaronson and Christiano Hidden Subspaces

- **Sample a random subspace A** of $\mathbb{F}_2^n$ with dimension n/2.
- The **dual subspace** is defined : $A^{\perp} = \left\{ b \in \mathbb{F}_2^n \mid \forall a \in A, a \cdot b = \sum_{i=1}^{n} a_i b_i \ mod\ 2 = 0 \right\}$
- A valid signature for **0:** $a \in A$
- A valid signature for **1:** $a \in A^{\perp}$

One time 1 bit Scheme

Membership Function

$$\chi_A(\alpha) = \begin{cases} 1 & \alpha \in A \\ 0 & otherwise \end{cases}$$

$$\chi_{A^*}(\alpha, p) = \begin{cases} \chi_A(\alpha) & p = 0 \\ \chi_{A^\perp}(\alpha) & p = 1 \end{cases}$$

One time 1 bit Scheme

Function key-gen(κ)

- n <- κ (will also work with smaller number of bits)
- Sample a random subspace A of n/2 dimension
- $pk \leftarrow O_{\chi_{A^*}}$ (An oracle access to the membership function)
- $sk \leftarrow <A>$ (a basis of A)

Function token-gen(sk)

- $return\ \tau = |A\rangle = \frac{1}{\sqrt{2^{n/2}}} \sum_{a \in A} |a\rangle$

One time 1 bit Scheme

Function sign(α, τ)

- Return the outcome of measuring $((H^{\otimes n})^{\alpha}\ \tau)$ in the computational basis

- α is a **single bit document**
- Recall that : $H^{\otimes n}|A\rangle = |A^{\perp}\rangle$
- The **output is classical**(vector from A or from dual A)
- The **token is consumed** in the process

One time 1 bit Scheme

Function verify(pk, α, sig)

- $return\ O_{\chi_{A^*}}(sig,\ \alpha)$
- Using the membership **oracle** we check:
  - If **α = 0** and sig is a **vector from A** then return true
  - If **α = 1** and sig is a **vector from dual A** then return true
  - **Else** return **false**

Security Intuition

- We will claim that it is **hard to sign** two distinct documents, given a signing token
- Once an attacker measures in the primal basis he can't sign another message

  (An attacker can **either** get an **element from A or** an **element from A dual**).
- The attacker cannot duplicate the signing token **(no cloning theorem)**

Attack On OT1

- Assume an **attacker measured** the token and **got an element from A**.
- The attacker can now create the **dual subspace for that vector**.
- This dual subspace contains 2^{n-1} vectors.
- **dual A contains** $2^{n/2}$ vectors(the dimension of dual A is n/2).
- **The probability to guess** a vector from dual A is $\frac{1}{2^{n/2-1}}$.
- Using Grover's algorithm we can find with high probability a vector from the dual subspace using $2^{n/4}$ queries.

Main Theorem

[BDS 16] Theorem 16

let A *be a uniformly random subspace of* $\mathbb{F}_2^n$, *and let* $\varepsilon > 0$ *be such that* $1/\varepsilon = o\left(2^{n/2}\right)$.

given one copy of $|A\rangle$ *and a quantum membership oracle for* A *and* $A^{\perp}$, *a counterfeiter needs* $\Omega\left(\sqrt{\varepsilon}\, 2^{n/4}\right)$ *queries to output a pair* (a, b) *such that* $a \in A$ *and* $b \in A^{\perp}$ *with probability at least* ε.

General Idea of Proof

- From the attack we showed before , we can get clear intuition about what the lower bound should be.
- In the classical setting , it would take $\Omega\left(2^{n/2}\right)$ to guess a vector with high probability.
- Using Grover's algorithm we only need root number of queries($\Omega\left(2^{n/4}\right)$).
- They proved that any algorithm which outputs a pair (a,b) where a in A and b in A dual(with probability at list 0.99) must take $\Omega\left(2^{n/4}\right)$ queries.

General Idea of Proof

- Then they showed that even with exponentially small probability ε where $1/_{\varepsilon} = o\left(2^{n/2}\right)$ any algorithm will still need to perform $\Omega\left(\sqrt{\varepsilon}\,2^{n/4}\right)$ queries.
- We can derive that directly from the constant probability.
- In the classical settings if we want to increase our probability by ε, we will need to perform ε queries(each query is independent).
- Using Grover's algorithm we only need to perform root number of queries.

Construction Overview

- ~~One Time tokenized scheme for 1 bit~~
- One Time tokenized scheme for restricted number of bits from 1 bit scheme
- One Time tokenized scheme for unrestricted number of bits from restricted scheme
- Tokenized Signature Scheme from unrestricted scheme.

Restricted Scheme

Why Do We Need It?

- In order to sign an arbitrary size document we will use the **hash and sign** paradigm.
- We **cannot map an arbitrary size document to 1 bit** document(hash collisions)
- A collision is formed when for $x \neq x'$, $h(x) = h(x')$
- **If an attacker can find a collision** in the hash, he can create two valid signatures using one signing token and **break the scheme**.
- Hash(a) = Hash(b) => sig(a) = sig(b)

Restricted Scheme - General Idea

- Use OT1 r times.
- The i-th token will correspond to the i-th bit in the document.
- The i-th pk will correspond to the i-th sig.

One Time restricted Scheme

Function key-gen(κ, r)

- for i = 0,...,r:
  - $pk_i, sk_i \leftarrow OT1.key-gen(\kappa)$
- Return pk, sk

One Time restricted Scheme

Function token-gen(sk)

- for i = 0,...,r:
  - $\tau_i \leftarrow OT1.token-gen(sk_i)$
- return $\boldsymbol{\tau}$

One Time restricted Scheme

Function sign(α, τ)

- for i = 0,...,r:
  - $sig_i \leftarrow OT1.sign(\alpha_i, \tau_i)$
- return sig

One Time restricted Scheme

Function verify(pk, α, sig)

- for i = 0,...,r:
  - return false if $OT1.verify(pk_i, \alpha_i, sig_i) = F$
- return true.

Reduction from r-bit to 1-bit

- We will show that **if OT1 is unforgeable then also OTR** is unforgeable.
- We will show that a successful attack on OTR **implies a successful attack on OTR1**, hence a **contradiction**.
- Given an Adv for OTR , we will **construct an Adv’** for OT1.

Reduction from r-bit to 1-bit

- The idea: **generate r-1 tokens** and then invoke Adv.
- Since each bit is signed individually then **signing two distinct r-bit messages implies two distinct 1-bit messages were signed**.

Reduction from r-bit to 1-bit

Adv'(pk, **τ**)

- Pick a random index j.
- $\tau'_j \leftarrow \tau$
- $pk'_j \leftarrow pk$
- For i=0,...,r without j
  - $pk'_i, sk'_i \leftarrow OT1.key-gen(\kappa)$
  - $\tau'_i \leftarrow OT1.token-gen\left(sk'_i\right)$
- $\left(\alpha_0, \alpha_1, \sigma_0, \sigma_1\right) \leftarrow Adv(pk', \tau')$
- $if\ \ \alpha_0[j] \neq \alpha_1[j]\ then\ return\ \left(\alpha_0[j],\ \alpha_1[j], \sigma_0[j], \sigma_1[j]\right)$

Reduction from r-bit to 1-bit

- Assume that the **success probability of Adv** is greater than f(**κ**) a **non-negligible** function.
- The success probability of Adv' is $Pr[OT1.verify_{2,pk}(Adv'(pk,\tau)) = T]$

Next we will show that $Pr[OT1.verify_{2,pk}(Adv'(pk,\tau)) = T] \geq non-negl(\kappa)$

Reduction from r-bit to 1-bit

- **Adv’** uses Adv but also require that **must differ in the j-th bit**.
- **Adv** require that the document will **differ** but it doesn’t have to be the j-th bit, just **some arbitrary bit.**
- $Pr[OT1.verify_{2,pk}(Adv'(pk,\tau))=T] \geq Pr\left[OTR.verify_{2,pk'}\left(Adv\left(pk',\tau_1 \otimes \ldots \otimes \tau_r\right)\right)=T \wedge \alpha_0[j] \neq \alpha_1[j]\right]$

Reduction from r-bit to 1-bit

- OTR sings an **r bit document**.
- We know that the messages must have **1 different bit**(or else verify won't accept).
- The **probability to guess** the differed bit is 1/r.
- $Pr\left[OTR.verify_{2,pk'}\left(Adv\left(pk',\tau_1\otimes\ldots\otimes\tau_r\right)\right)=T\wedge\alpha_0[j]\neq\alpha_1[j]\right]\geq\frac{1}{r}\cdot Pr\left[OTR.verify_{2,pk'}\left(Adv\left(pk',\tau_1\otimes\ldots\otimes\tau_r\right)\right)=T\right]$

Reduction from r-bit to 1-bit

- We assumed that the **success probability of Adv is non-negl(κ)**
- $\frac{1}{r} \cdot Pr\left[OTR.verify_{2,pk'}\left(Adv\left(pk', \tau_1 \otimes \ldots \otimes \tau_r\right)\right) = T\right] \geq \frac{f(\kappa)}{r} \geq non-negl(\kappa)$
- We proved that if OT1 is secure then also OTR.
- We didn't assume anything about OT1.

Construction Overview

- ~~One Time tokenized scheme for 1 bit~~
- ~~One Time tokenized scheme for restricted number of bits from 1 bit scheme~~
- One Time tokenized scheme for unrestricted number of bits from restricted scheme
- Tokenized Signature Scheme from unrestricted scheme

Moving to unrestricted Scheme

- **Extend the previous** schemes to support a document of arbitrary size.
- Hash the document down to r bits, and then to sign the hash (**hash-and-sign**).
- Recall that : A **collision is formed** when for $x \neq x'$, $h(x) = h(x')$. Also, **If an attack can find collision** in the hash function he can break the scheme.
- Require that the hash function is **collision resistant** against a QPT Adv.

Collision Resistant Hash Function

- Let $r(\kappa) : \mathbb{N} \rightarrow \mathbb{N}$
- $\left\{h_s : \{0,1\}^* \rightarrow \{0,1\}^{r(|s|)}\right\}_{s \in \{0,1\}^*}$
- We require an **efficient evaluation** - given s and x returns $h_s(x)$ in polynomial time
- We also require that for any QPT adversary which given the index of the hash(s) output a **collision in a negligible probability**: $Pr\left[Adv(s) \text{ is a collision under } h_s\right] \leq negl(\kappa)$
- The **length of resulting signatures** only depends on the length s, and is **independent of the document** being signed.

One Time unrestricted Scheme

Function key-gen(κ)

- s <- index(κ) (s is the index of a hash function)
- $sk', pk' \leftarrow OTR.key-gen(\kappa, r(\kappa))$
- $sk \leftarrow (sk', s), pk \leftarrow (pk', s)$

One Time unrestricted Scheme

Function token-gen(sk)

- $\tau' \leftarrow OTR.token-gen(sk')$
- $\tau \leftarrow (s, \tau')$

One Time unrestricted Scheme

Function sign(α, τ)

- $return\ OTR.sign\left(h_s(\alpha),\ \tau'\right)$

Function verify(pk, α, sig)

- $return\ OTR.verify_{pk'}\left(h_s(\alpha),\ sig\right)$

Construction Overview

- ~~One Time tokenized signature scheme for 1 bit~~
- ~~One Time tokenized signature scheme for restricted number of bits from 1 bit scheme~~
- ~~One Time tokenized signature scheme for unrestricted number of bits from restricted scheme~~
- Tokenized Signature Scheme from unrestricted scheme

Tokenized Signature Scheme

Moving from one time schemes to a scheme that can support an arbitrary number of tokens.

Security Intuition:

- We enforce that **every token will have a new pk and sk**
- **the tokens are independent** , an attacker cannot use the generated tokens to gain any more knowledge.

Tokenized Signature Scheme

Why do we need DS?

- An attacker can still create is own tokens and **cheat the verifier**.
- We will use our digital signature scheme to **sign the OT public key** generated by OT.key-gen.
- By doing that, the verifier can now **verify the identity of the token creator**.

Tokenized Signature Scheme

Function key-gen(κ)

- $(pk, sk) \leftarrow DS.key-gen(\kappa)$, where DS is a digital signature scheme

Function token-gen(sk)

- $OT.pk, OT.sk \leftarrow OT.key-gen(\kappa)$
- $\tau' \leftarrow OT.token-gen(OT.sk)$
- $return\ \tau = (OT.pk, DS.sign(sk, OT.pk), \tau')$

Tokenized Signature Scheme

Function sign(α, τ)

- $sig' \leftarrow OT.sign(\alpha, \tau')$
- $return\ (OT.pk, DS.sign(sk, OT.pk), sig')$

Function verify(pk, α, sig)

- $return\ DS.verify_{pk}(OT.pk, DS.sign(sk, OT.pk)) \wedge OT.verify_{OT.pk}(\alpha, sig')$

What other functionalities can we think of?

Revocability

Motivation:

- **A contract** between A and B came **to an end**
- As part of the contract **A gave B some tokens**
- How can A make sure that **B won’t use** them in the future?

Revocability

- A revocable TS scheme adds a **fifth QPT algorithm** : revoke.
- Revoke **verify** a given token **and consume** it in the process.
- Revoke only requires the **public key**.
- Correctness: $Pr\left[revoke_{pk}(\tau) = T\right] = 1$
- Security: $Pr\left[verify_{t,pk}\left(\alpha_1, sig_1, \ldots, a_t, sig_t\right) = T \bigwedge revoke_{l-t+1,pk}(\sigma) = T\right] \leq negl(k)$

Tokenized Signature Scheme - Testability

- A testable TS scheme adds a **fifth QPT algorithm** : verify-token.
- Verify-token verify a given token **without consuming** it.
- A testable TS scheme can be used as a **quantum money** scheme.
- Correctness: $Pr\left[verify-token_{pk}(\tau) \ =(T,\tau)\right]=1$
- Security: $Pr\left[verify_{pk}(\alpha,sign(\alpha,\sigma))=T \mid (T,\sigma)\leftarrow verify-token_{pk}(\tau)\right]\geq 1-negl(k)$

  $Pr\left[verify-token_{pk}(\sigma)=T \mid (T,\sigma)\leftarrow verify-token_{pk}(\tau)\right]\geq 1-negl(k)$

Summary

- Motivation
- Defining Cryptographic Primitives
    - Digital Signature
    - Quantum Tokens
- Constructions
    - One Time tokenized signature scheme for 1 bit
    - One Time tokenized signature scheme for restricted number of bits
    - One Time tokenized signature scheme for unrestricted number of bits
    - Tokenized Signature Scheme

Open Questions

- Is there a way to make the token sign a **specific type of documents**? With or without the knowledge of the Signer?

- The output of sign is classical , which means an attacker can **duplicate the signature** a multiple number of times, is there any way we can prevent that kind of duplication? Without the verifier holding some database of used tokens.

THE END