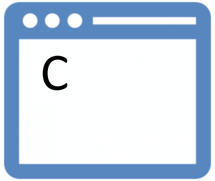
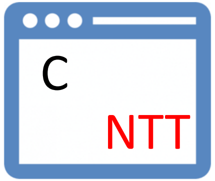
Watermarking PRF against Quantum Adversaries

Fuyuki Kitagawa (NTT Corporation)
Ryo Nishimaki (NTT Corporation)

Software watermarking



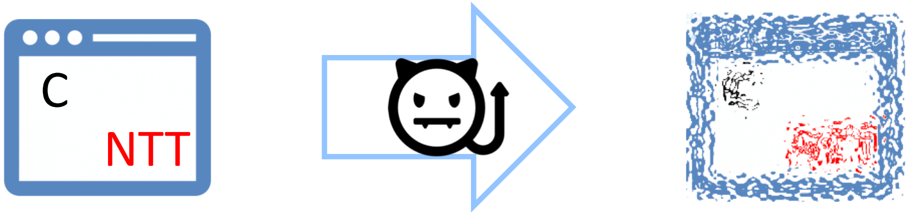
Software watermarking



We can embed a mark
into a program

Software watermarking

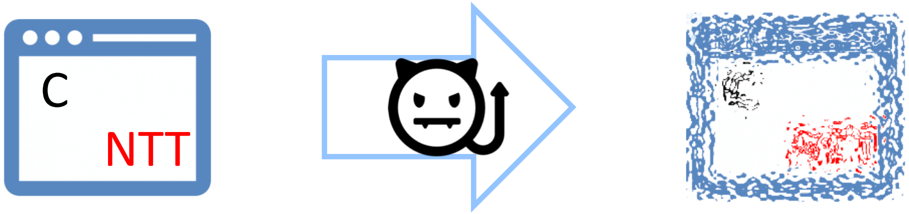
NTT



We can embed a mark into a program

If the mark is removed the program is destroyed

Software watermarking



We can embed a mark into a program

If the mark is removed the program is destroyed

- The purpose: proving ownership, preventing illegal copies, and so on

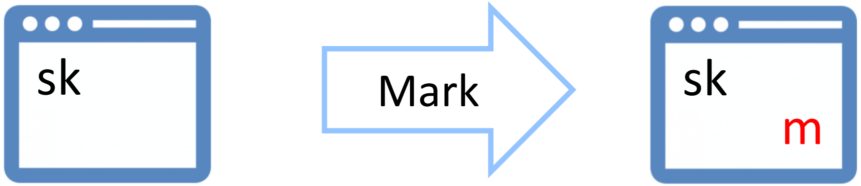
Watermarking crypto programs

- Software watermarking can deal with only unlearnable programs

  - If a program is learnable, the mark is removed by learning original program

Watermarking crypto programs

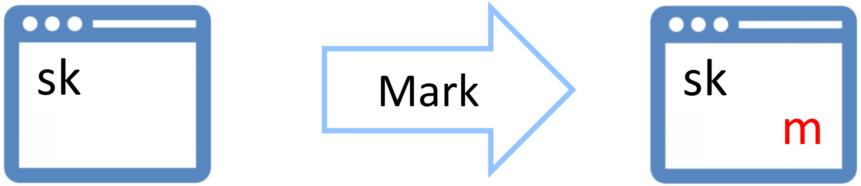
- Software watermarking can deal with only unlearnable programs
  - If a program is learnable, the mark is removed by learning original program
- Previous works have focused on watermarking crypto programs



  - Especially, most of them studied watermarking PRF
    - Simplest, but sufficient for many other crypto primitives

Watermarking crypto programs

- Software watermarking can deal with only unlearnable programs
  - If a program is learnable, the mark is removed by learning original program
- Previous works have focused on watermarking crypto programs

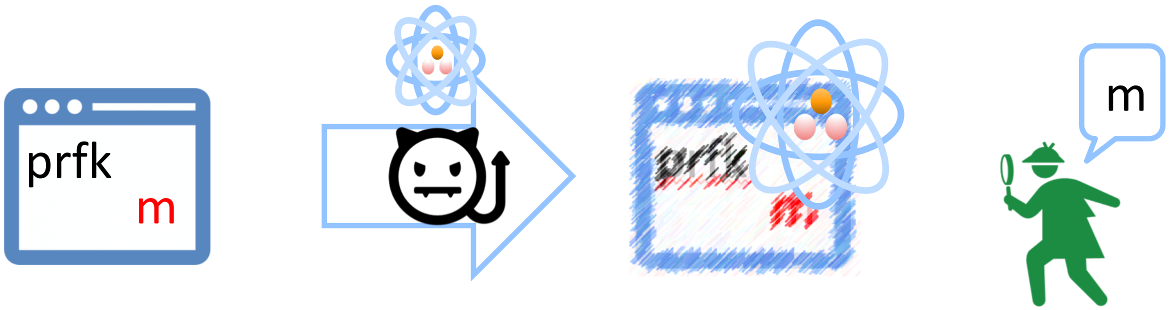


  - Especially, most of them studied watermarking PRF
    - Simplest, but sufficient for many other crypto primitives
- Application to quantum crypto [KNY21,ALLZZ21]
  - By combining with quantum money, we can construct secure software leasing

This work

NTT

- Watermarking PRF against quantum adversaries



Our results

NTT

1. We define watermarking PRF against quantum adversaries

2. We construct watermarking PRF against quantum adversaries

Our results

1. We define watermarking PRF against quantum adversaries
   - We define unremovability against adversaries who output quantum program
2. We construct watermarking PRF against quantum adversaries

Our results

NTT

1. We define watermarking PRF against quantum adversaries
    - We define unremovability against adversaries who output quantum program
2. We construct watermarking PRF against quantum adversaries
    - Secret extractable scheme based on LWE
    - Public extractable scheme based on IO

Our results

NTT

1. We define watermarking PRF against quantum adversaries

   - We define unremovability against adversaries who output quantum program

2. We construct watermarking PRF against quantum adversaries

   - Secret extractable scheme based on LWE
   - Public extractable scheme based on IO

   - Our construction methodology is highly general
     and can be extended to watermarking other primitives such as PKE

Technical overview

- The biggest issue is that quantum programs are stateful programs

  - It was pointed out by Zhandry [Zha20] in the context of traitor tracing
  - Classical traitor tracing/watermarking assumes pirate programs are stateless
    - It is reasonable since we can rewind pirate programs
  - It is impossible to rewind quantum program

Technical overview

- The biggest issue is that quantum programs are stateful programs

  - It was pointed out by Zhandry [Zha20] in the context of traitor tracing
  - Classical traitor tracing/watermarking assumes pirate programs are stateless
    - It is reasonable since we can rewind pirate programs
  - It is impossible to rewind quantum program

- We need to define unremovability taking it into consideration

- We need to construct our scheme taking it into consideration

Technical overview

- The biggest issue is that quantum programs are stateful programs

  - It was pointed out by Zhandry [Zha20] in the context of traitor tracing
  - Classical traitor tracing/watermarking assumes pirate programs are stateless
    - It is reasonable since we can rewind pirate programs
  - It is impossible to rewind quantum program

- We need to define unremovability taking it into consideration

  - We can use Zhandry [Zha20]'s technique

- We need to construct our scheme taking it into consideration

Technical overview

NTT

- The biggest issue is that quantum programs are stateful programs

  - It was pointed out by Zhandry [Zha20] in the context of traitor tracing
  - Classical traitor tracing/watermarking assumes pirate programs are stateless
    - It is reasonable since we can rewind pirate programs
  - It is impossible to rewind quantum program

- We need to define unremovability taking it into consideration

  - We can use Zhandry [Zha20]'s technique

- We need to construct our scheme taking it into consideration

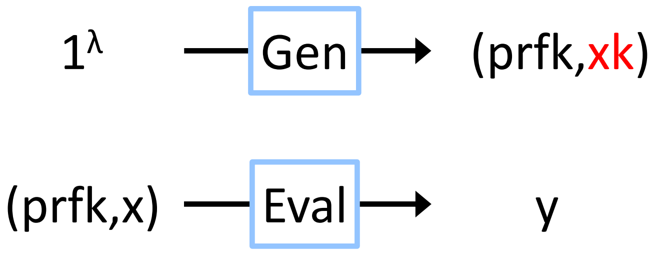
  - We cannot use Zhandry [Zha20]'s technique
  - We propose new extraction method

Syntax and functionality preserving

- Essentially the same as (single-key) traceable PRF [GKWW21]
- Consists of (Gen,Eval,Mark,Ext)

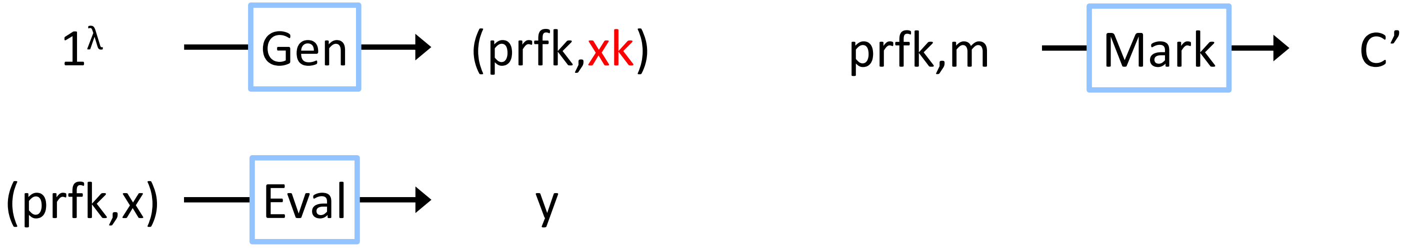
Syntax and functionality preserving

- Essentially the same as (single-key) traceable PRF [GKWW21]
- Consists of (Gen,Eval,Mark,Ext)



Syntax and functionality preserving

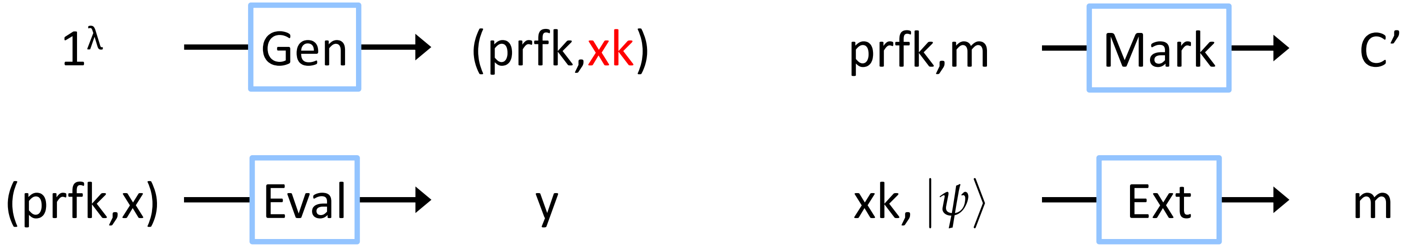
- Essentially the same as (single-key) traceable PRF [GKWW21]
- Consists of (Gen,Eval,Mark,Ext)



Syntax and functionality preserving

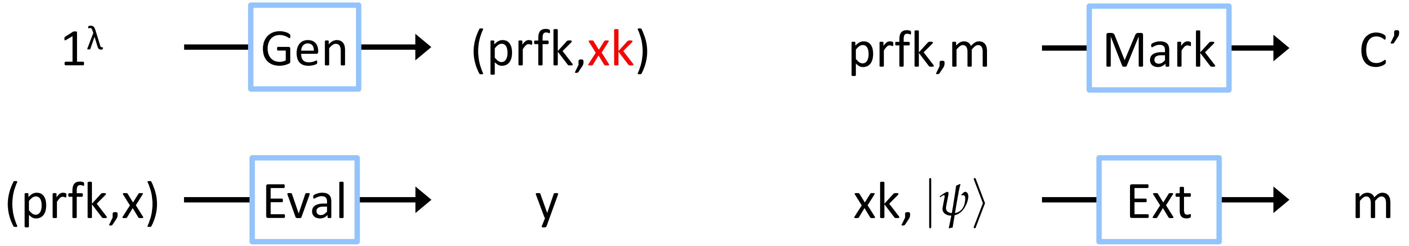
NTT

- Essentially the same as (single-key) traceable PRF [GKWW21]
- Consists of (Gen,Eval,Mark,Ext)



Syntax and functionality preserving

- Essentially the same as (single-key) traceable PRF [GKWW21]
- Consists of (Gen,Eval,Mark,Ext)



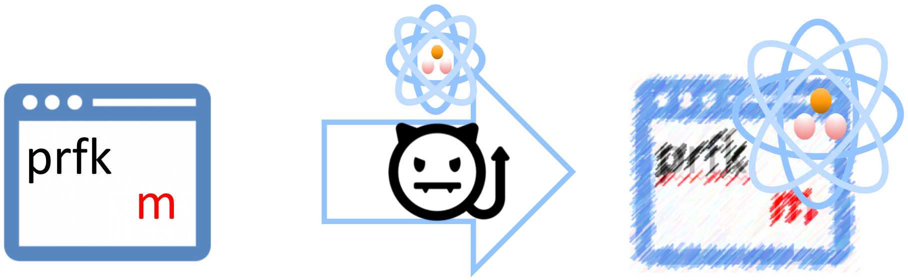
- Functionality preserving:

  C’(x) = Eval(prfk,x) for almost all inputs x

Unremovability

NTT

- The definition is roughly as follows

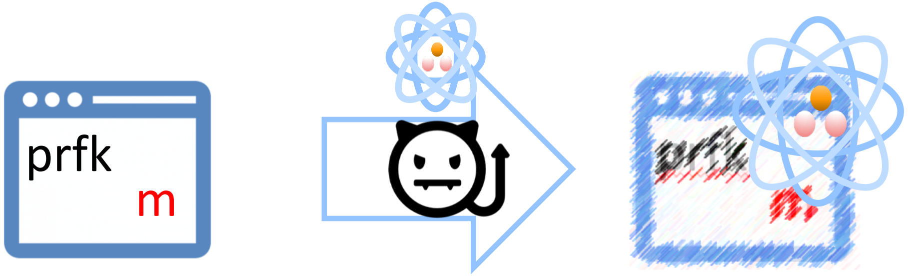


C’ = Mark(prfk,m)

$|\psi\rangle$

Unremovability

- The definition is roughly as follows



C’ = Mark(prfk,m) $|\psi\rangle$

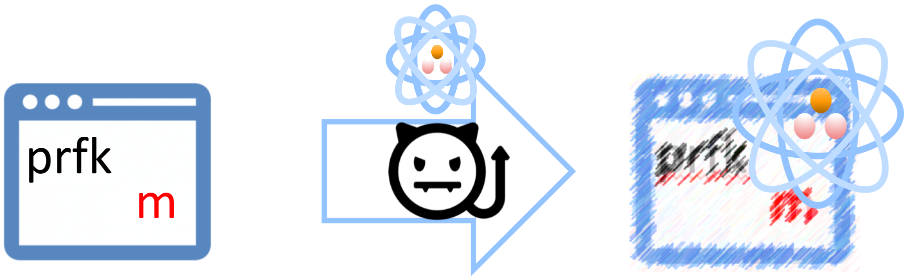
No adversary can generate $|\psi\rangle$ s.t.

- $|\psi\rangle$ is “good” in the sense that its functionality is close to Eval(prfk,·)
- Ext fails to extract m

Unremovability

NTT

- The definition is roughly as follows



C' = Mark(prfk,m) $|\psi\rangle$

No adversary can generate $|\psi\rangle$ s.t.

- $|\psi\rangle$ is "good" in the sense that its functionality is close to Eval(prfk,·)
- Ext fails to extract m

- To make the definition rigorous, we have to define "good" more concretely

How to define "good" quantum program?

- Define as quantum programs breaking weak PRF security [GKWW21]

How to define “good” quantum program?

- Define as quantum programs breaking weak PRF security [GKWW21]
- We have to take the stateful nature of them into consideration

How to define "good" quantum program?

- Define as quantum programs breaking weak PRF security [GKWW21]

- We have to take the stateful nature of them into consideration

  - We have to measure the advantage for breaking weak PRF security

How to define “good” quantum program?

- Define as quantum programs breaking weak PRF security [GKWW21]

- We have to take the stateful nature of them into consideration

  - We have to measure the advantage for breaking weak PRF security

  - The measurement itself might destroy the quantum program

How to define “good” quantum program?

- Define as quantum programs breaking weak PRF security [GKWW21]

- We have to take the stateful nature of them into consideration

  - We have to measure the advantage for breaking weak PRF security

  - The measurement itself might destroy the quantum program

  - If we once confirm a quantum program is a good program,
    the post measurement state might not be a good program

How to define “good” quantum program?

NTT

- Define as quantum programs breaking weak PRF security [GKWW21]

- We have to take the stateful nature of them into consideration

  - We have to measure the advantage for breaking weak PRF security

  - The measurement itself might destroy the quantum program

  - If we once confirm a quantum program is a good program,
    the post measurement state might not be a good program

- We use the notion of Live by Zhandry[Zha20] as the notion of “good”

How to define “good” quantum program?

- Define as quantum programs breaking weak PRF security [GKWW21]

- We have to take the stateful nature of them into consideration

  - We have to measure the advantage for breaking weak PRF security

  - The measurement itself might destroy the quantum program

  - If we once confirm a quantum program is a good program,
    the post measurement state might not be a good program

- We use the notion of Live by Zhandry[Zha20] as the notion of “good”

  - It is defined by using projective implementation (ProjImp)

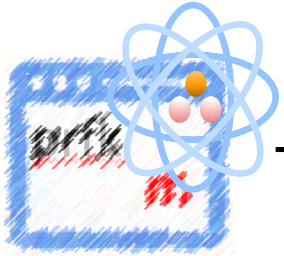
On quantum programs

- Success probability for breaking weak PRF is defined as follows

D_{wprf}: b ← {0,1}, x ← Dom, y_0 ← Ren, y_1 ← Eval(prfk,x), output (b,x,y_b)

Success probability is the probability that

(b,x,y_b) ← D_{wprf}

(x,y_b) ⟶  ⟶ b

$|\psi\rangle$

On quantum programs

NTT

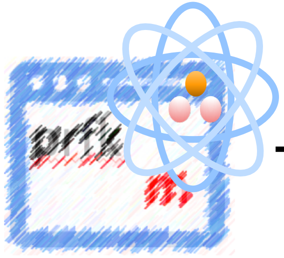
- Success probability for breaking weak PRF is defined as follows

  D_{wprf}: b ← {0,1}, x ← Dom, y_0 ← Ren, y_1 ← Eval(prfk,x), output (b,x,y_b)

  Success probability is the probability that

  (b,x,y_b) ← D_{wprf}

  (x,y_b) ⟶



  ⟶ b

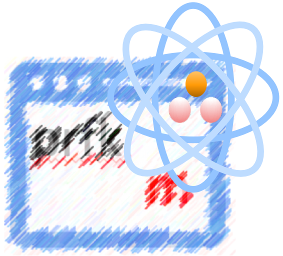
  $|\psi\rangle$

- $|\psi\rangle$ can be seen as super-position of programs with different success probabilities w.r.t D_{wprf}
  - $|\psi\rangle = \sum_p \alpha_p |\psi_p\rangle$, where $|\psi_p\rangle$ has success probability p w.r.t D_{wprf} and $\sum_p \alpha_p^2 = 1$

ProjImp and live quantum programs [Zha20]

NTT

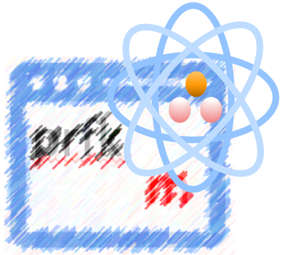
- Quantum program can be seen as a super-position of many programs
- ProjImp measures the success probability of one of them
- Concretely, it is defined as follows



$$|\psi\rangle = \sum_p \alpha_p \, |\psi_p\rangle$$

ProjImp and live quantum programs [Zha20]

- Quantum program can be seen as a super-position of many programs
- ProjImp measures the success probability of one of them
- Concretely, it is defined as follows



$|\psi\rangle = \sum_p \alpha_p |\psi_p\rangle$

Outcome p

with probability α_p^2

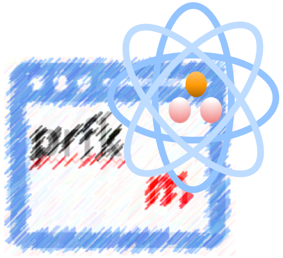
ProjImp(D_{wprf})

ProjImp and live quantum programs [Zha20]

- Quantum program can be seen as a super-position of many programs
- ProjImp measures the success probability of one of them
- Concretely, it is defined as follows

Success probability p

$|\psi_p\rangle$



$|\psi\rangle = \sum_p \alpha_p |\psi_p\rangle$

Outcome p

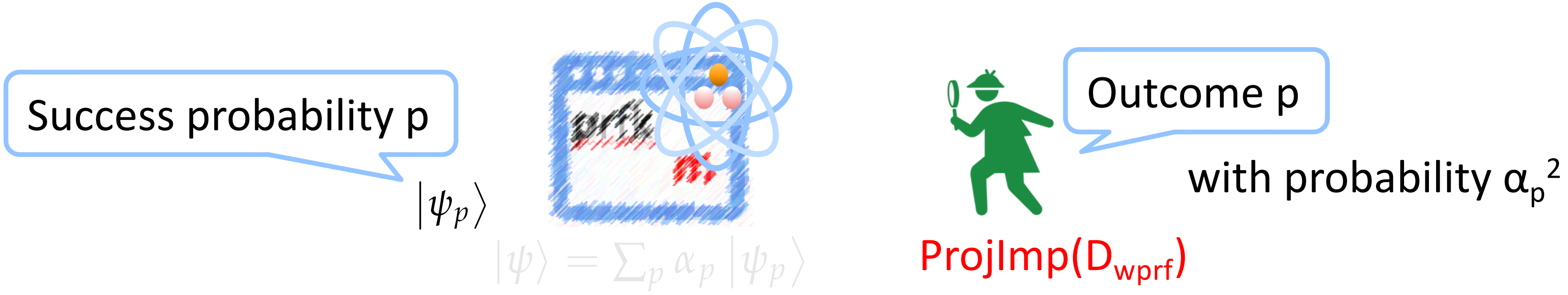
with probability α_p^2

ProjImp(D_{wprf})

ProjImp and live quantum programs [Zha20]

NTT

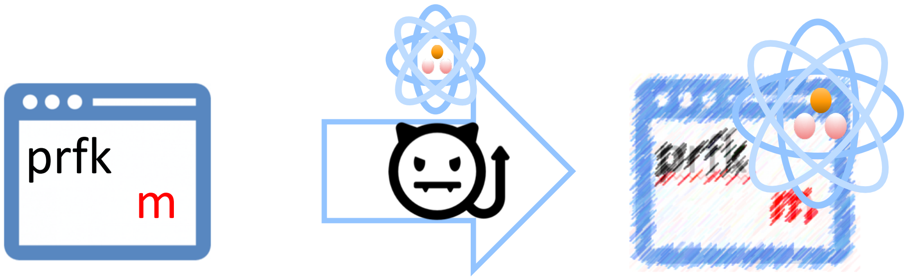
- Quantum program can be seen as a super-position of many programs
- ProjImp measures the success probability of one of them
- Concretely, it is defined as follows



- $|\psi\rangle$ is Live if the result p of ProjImp(D_{wprf}) is significantly greater than 1/2
  - For classical programs, it is the same as classical "good" [GKWW21]

Unremovability

NTT



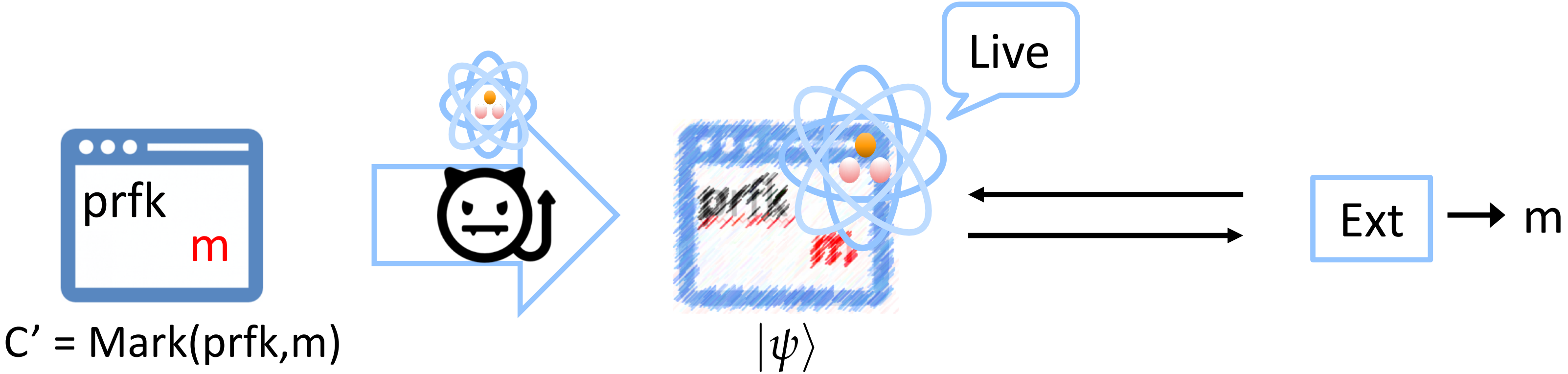
C’ = Mark(prfk,m) $|\psi\rangle$

No adversary can generate $|\psi\rangle$ s.t.

- $|\psi\rangle$ is a Live quantum program
- Ext fails to extract m

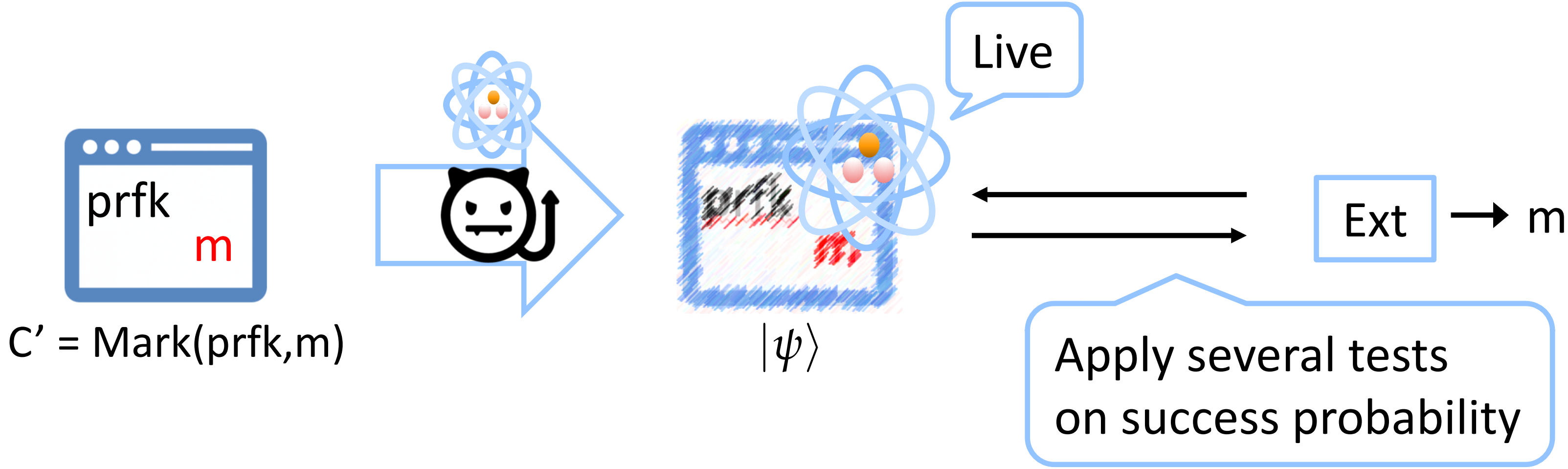
How to extract mark?

- Our goal is



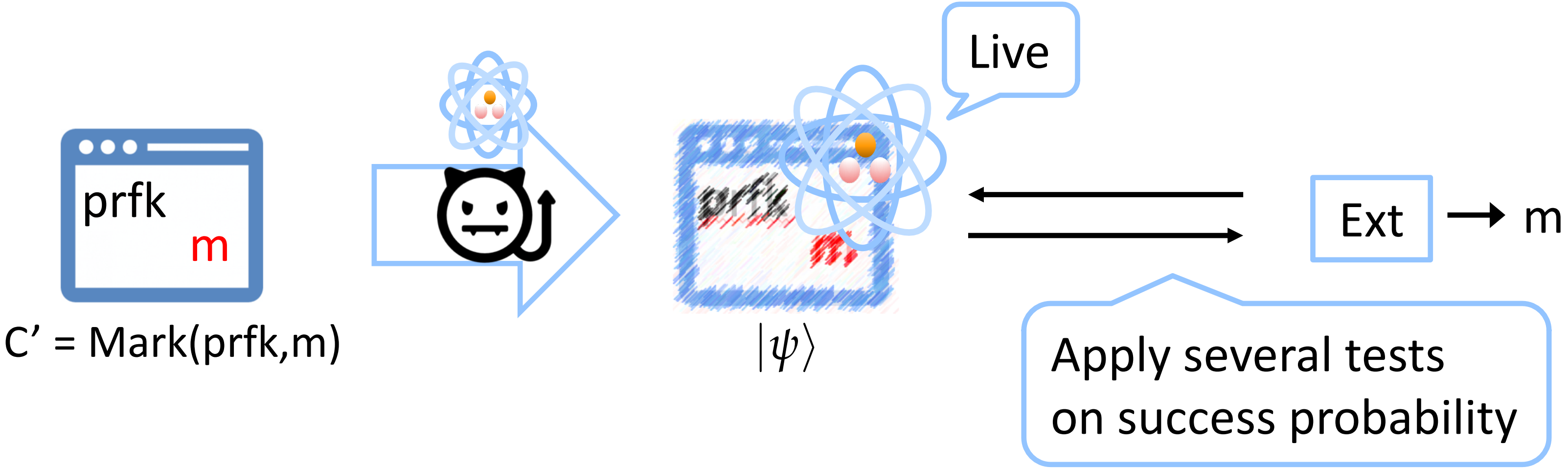
How to extract mark?

- Our goal is



How to extract mark?

- Our goal is



- The set of applicable tests is highly limited compared to classical case

  - Due to the stateful nature, a test can destroy the quantum program

Difficulty

- In watermarking, an embedded mark is chosen from super-poly size set

Difficulty

- In watermarking, an embedded mark is chosen from super-poly size set

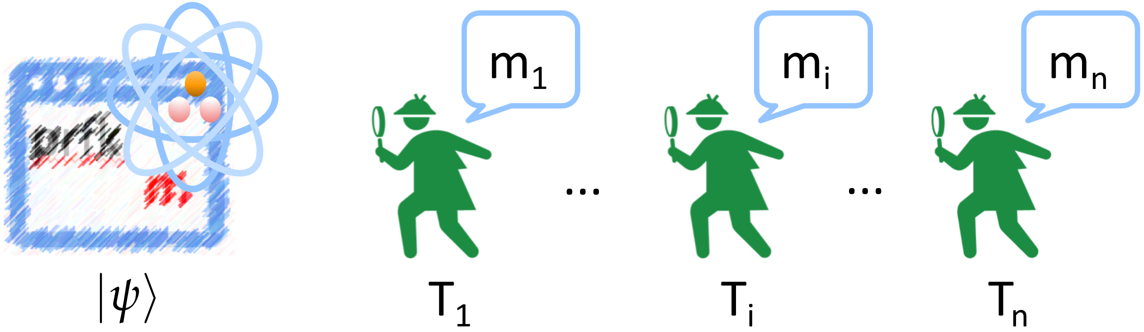
  - Extraction of mark is bit-by-bit manner

Difficulty

- In watermarking, an embedded mark is chosen from super-poly size set

  - Extraction of mark is bit-by-bit manner

- We need to realize a test T_i for every i s.t.



- T_i can be used to extract the i-th bit of the mark

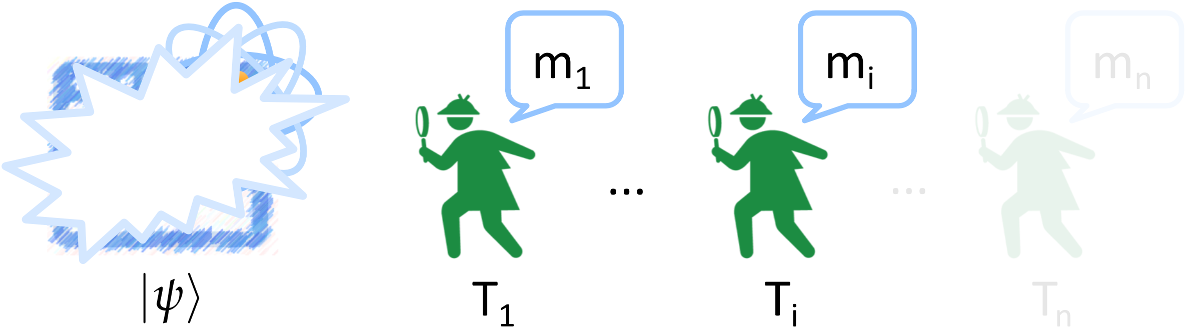
- T_i does not destroy the quantum program

Difficulty

- In watermarking, an embedded mark is chosen from super-poly size set

  - Extraction of mark is bit-by-bit manner

- We need to realize a test T_i for every i s.t.



  - T_i can be used to extract the i-th bit of the mark

  - T_i does not destroy the quantum program

Our idea

NTT

- We use reverse projective property of ProjImp
- Let D_{fail} be the distribution generates (b,x,y) ← D_{wprf} and outputs (1-b,x,y)

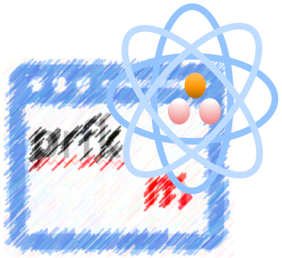
  - ProjImp(D_{fail}) measures failure probability

Our idea

NTT

- We use reverse projective property of ProjImp
- Let D_{fail} be the distribution generates (b,x,y) ← D_{wprf} and outputs (1-b,x,y)

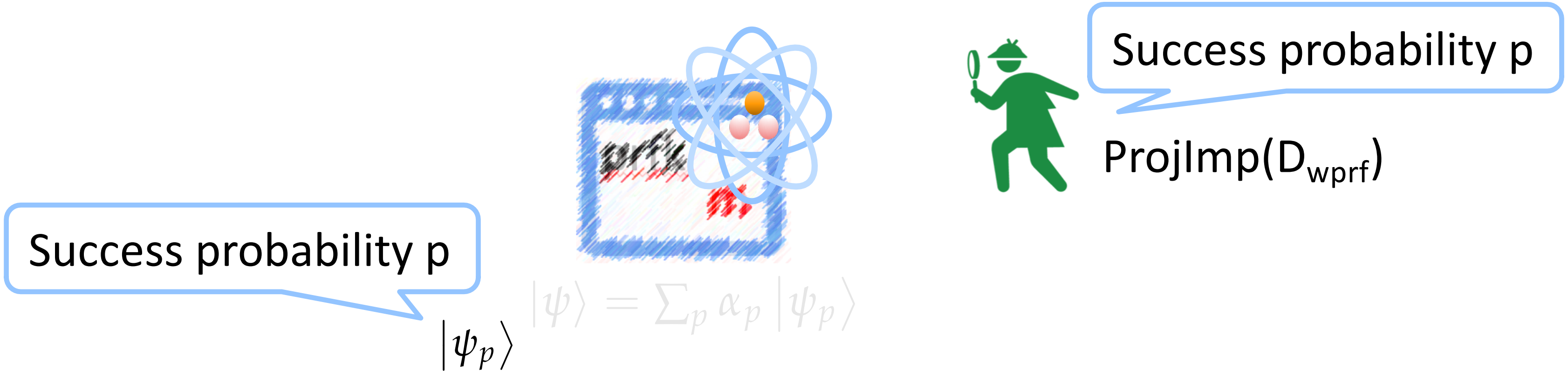
  - ProjImp(D_{fail}) measures failure probability



$$|\psi\rangle = \sum_p \alpha_p |\psi_p\rangle$$

Our idea

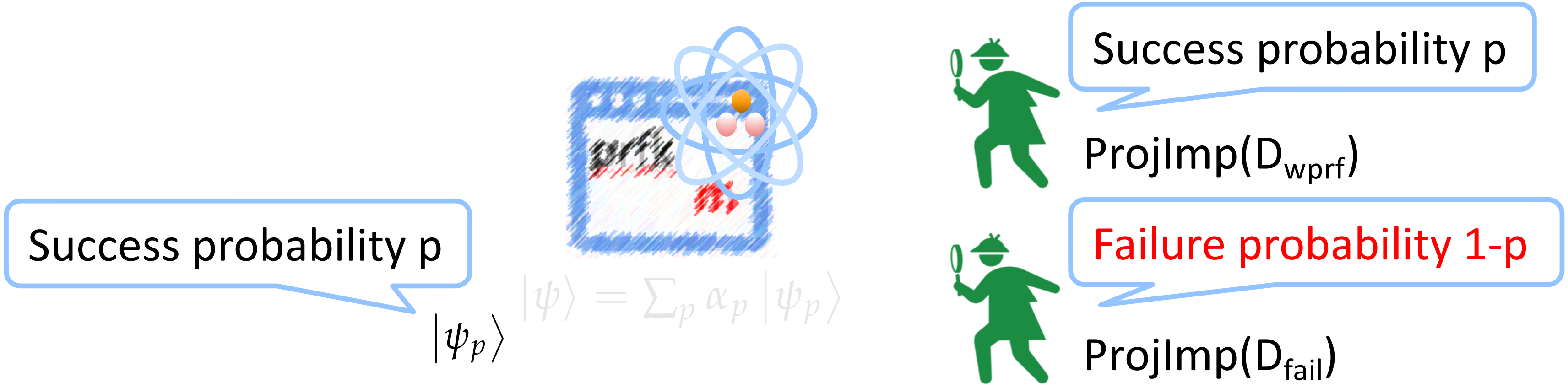
- We use reverse projective property of ProjImp
- Let D_{fail} be the distribution generates (b,x,y) ← D_{wprf} and outputs (1-b,x,y)
  - ProjImp(D_{fail}) measures failure probability



Our idea

- We use reverse projective property of ProjImp
- Let D_{fail} be the distribution generates (b,x,y) ← D_{wprf} and outputs (1-b,x,y)

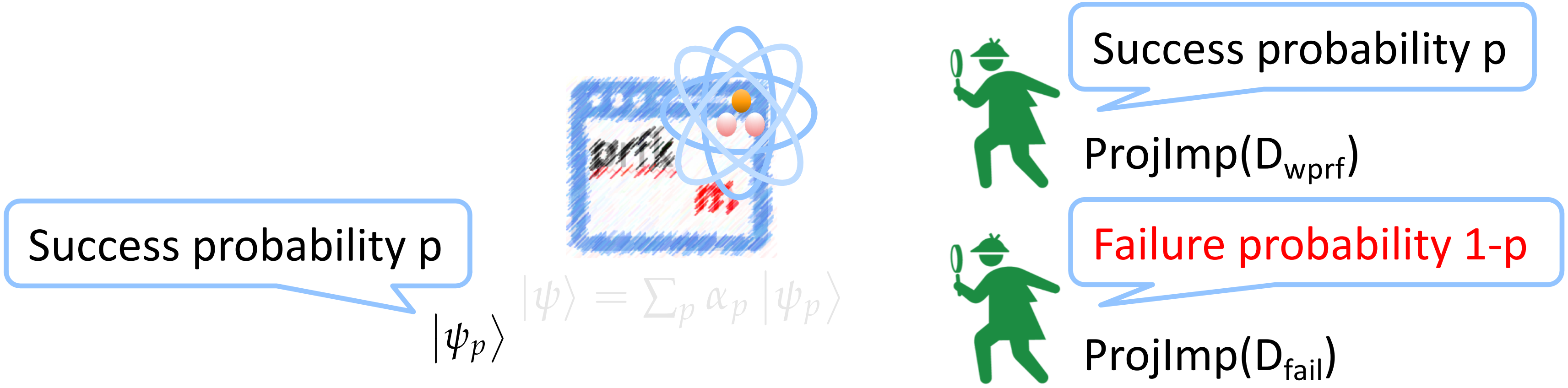
  - ProjImp(D_{fail}) measures failure probability



Our idea

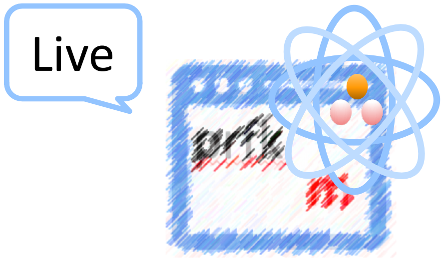
- We use reverse projective property of ProjImp
- Let D_{fail} be the distribution generates (b,x,y) ← D_{wprf} and outputs (1-b,x,y)

  - ProjImp(D_{fail}) measures failure probability



- ProjImp(D_{wprf}) and ProjImp(D_{fail}) consist of the same set of operators and the only difference is labels of them

Key fact



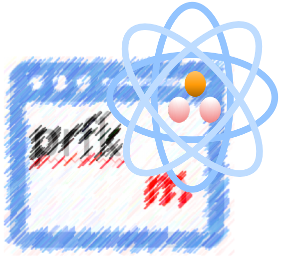
Key fact

Outcome 1/2+ε

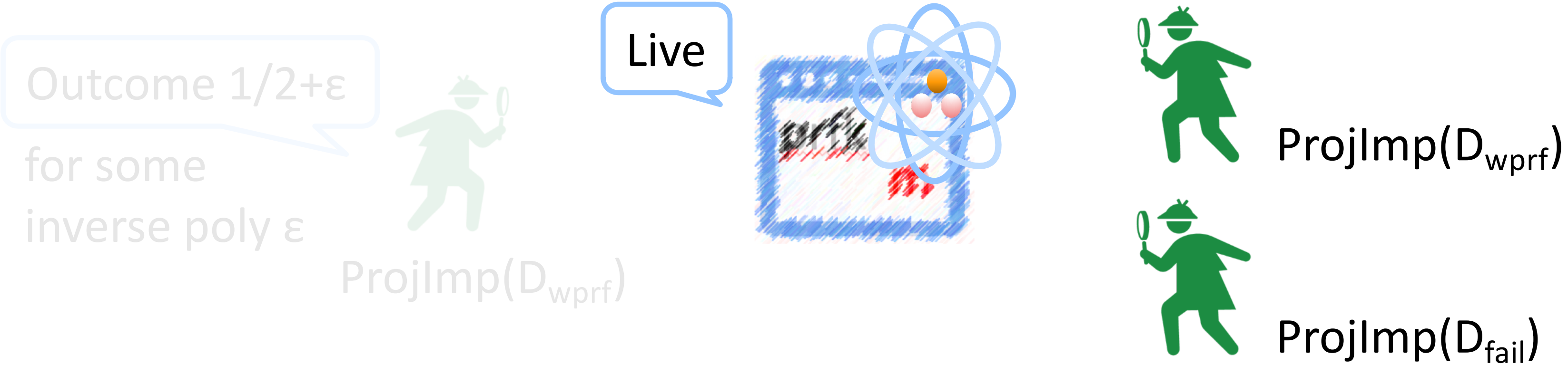
for some inverse poly ε

ProjImp(D_{wprf})

Live



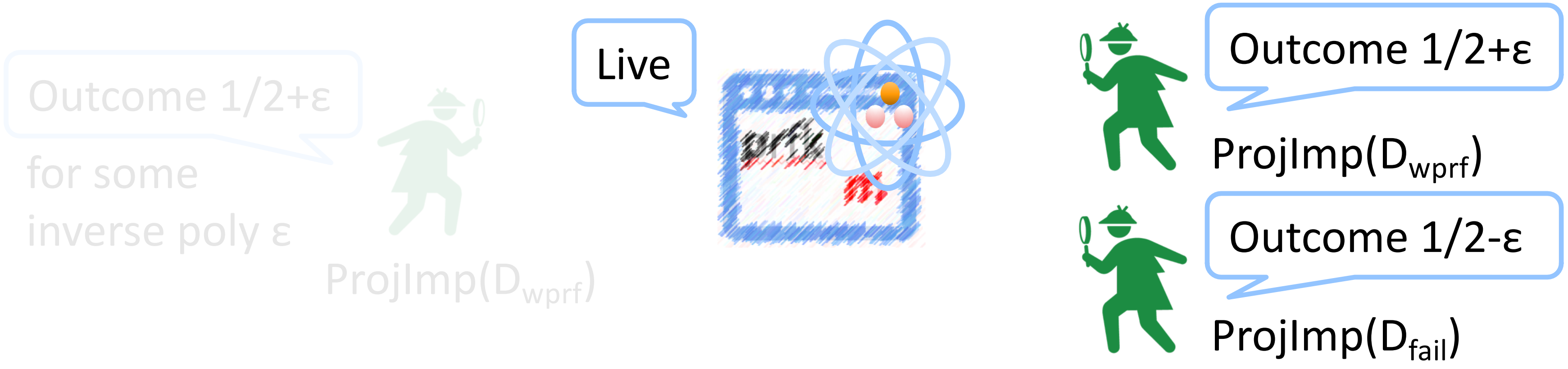
Key fact


Outcome 1/2+ε
for some
inverse poly ε
ProjImp(D_{wprf})
Live
ProjImp(D_{wprf})
ProjImp(D_{fail})

Key fact


Outcome 1/2+ε
for some
inverse poly ε
ProjImp(D_{wprf})
Live
prf$_k$
Outcome 1/2+ε
ProjImp(D_{wprf})
ProjImp(D_{fail})

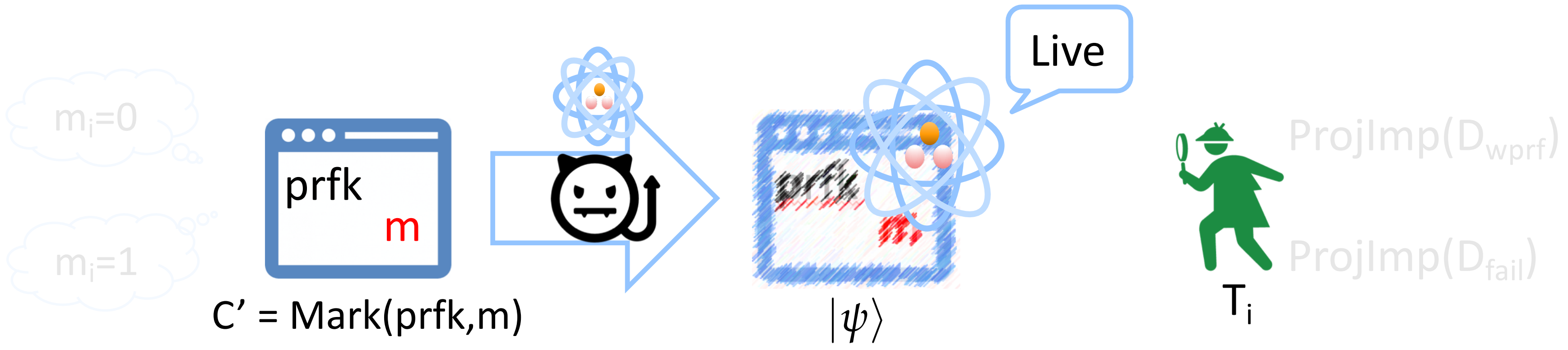
Key fact



Outcome 1/2+ε
for some
inverse poly ε
ProjImp(D_wprf)
Live
Outcome 1/2+ε
ProjImp(D_wprf)
Outcome 1/2-ε
ProjImp(D_fail)

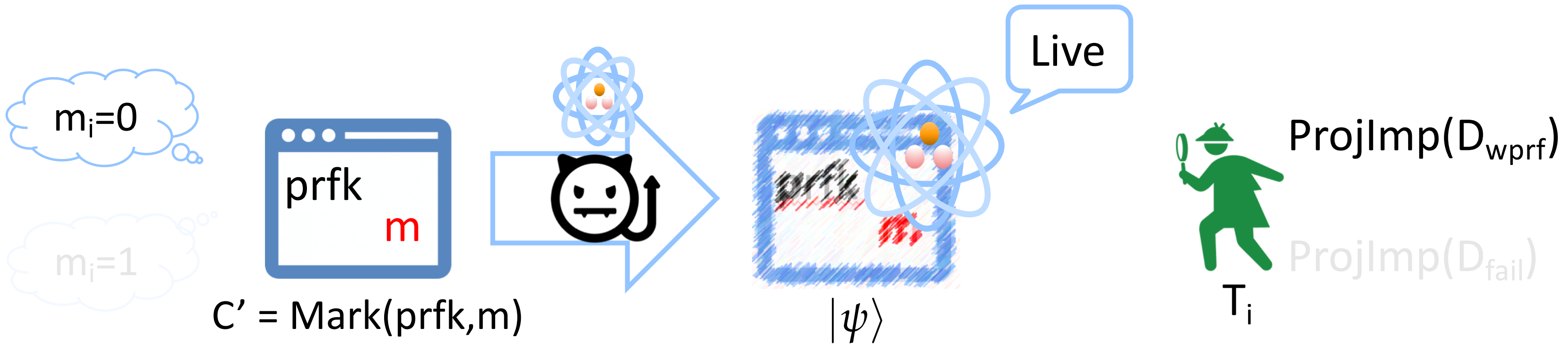
Our extraction method

- Our extraction uses T_i with the following properties for every i



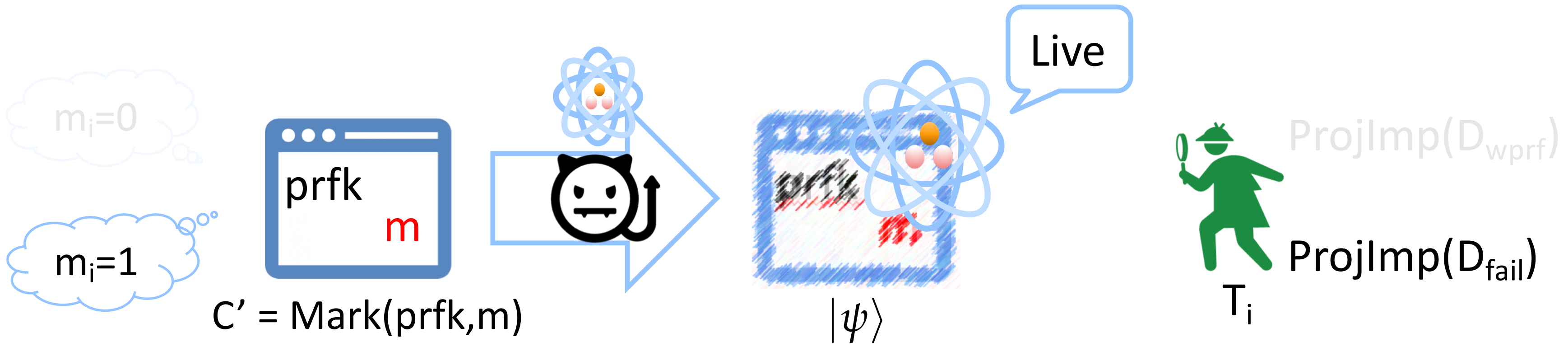
Our extraction method

- Our extraction uses T_i with the following properties for every i



Our extraction method

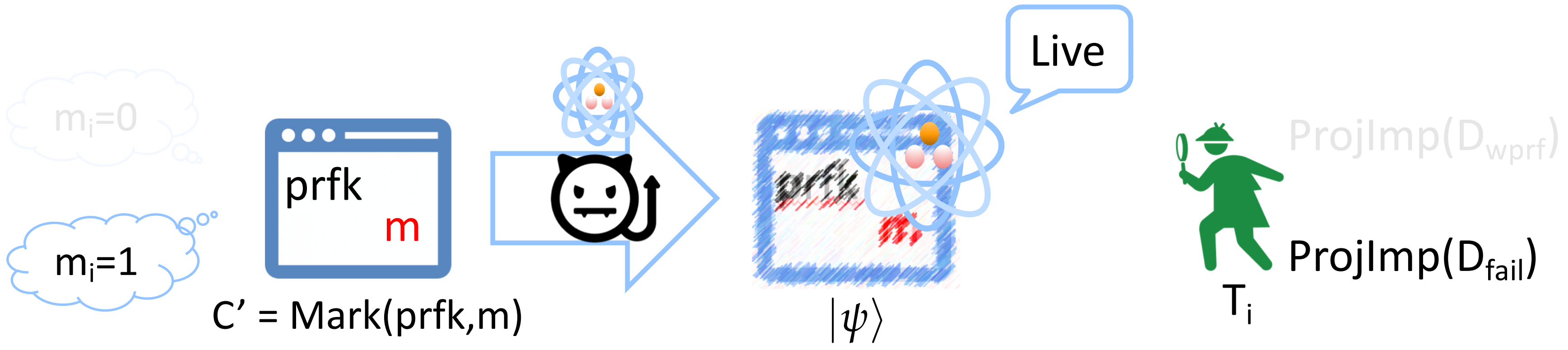
- Our extraction uses T_i with the following properties for every i



Our extraction method

NTT

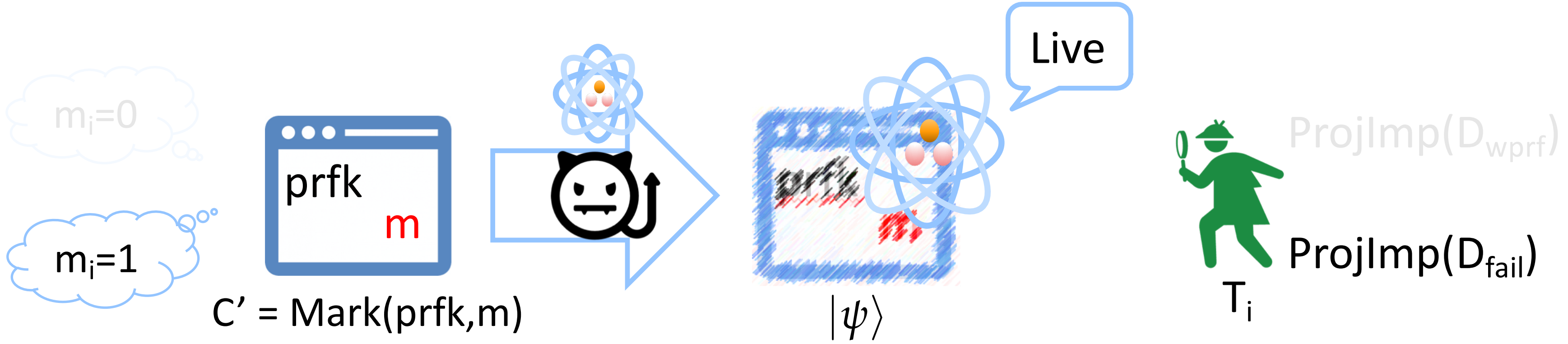
- Our extraction uses T_i with the following properties for every i



- From our key fact
  - The outcome of T_i is 1/2+ε if m_i=0 and 1/2-ε if m_i=1

Our extraction method

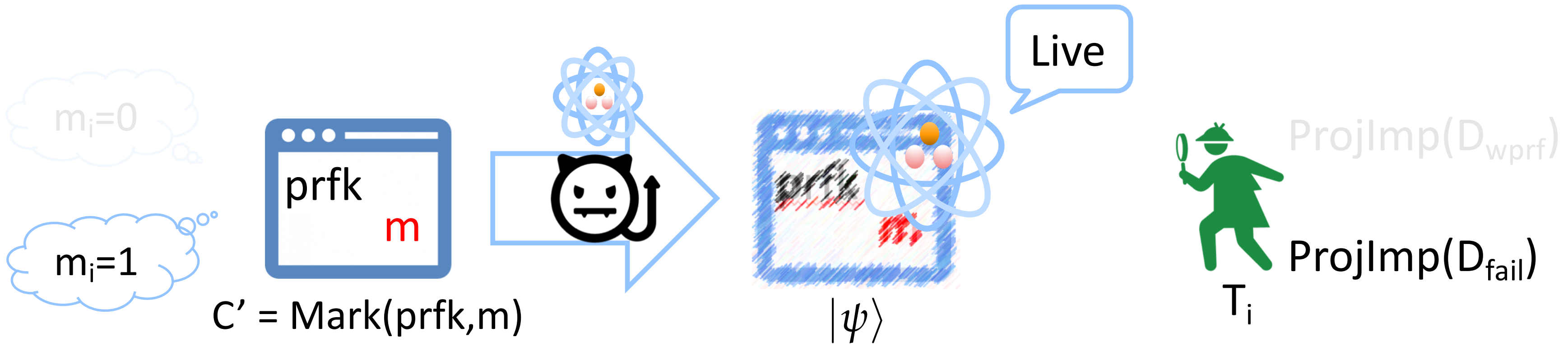
- Our extraction uses T_i with the following properties for every i



- From our key fact
  - The outcome of T_i is 1/2+ε if m_i=0 and 1/2-ε if m_i=1
  - T_i does not destroy the quantum program

Our extraction method

- Our extraction uses T_i with the following properties for every i



- From our key fact
  - The outcome of T_i is 1/2+ε if m_i=0 and 1/2-ε if m_i=1
  - T_i does not destroy the quantum program
- Our extraction method correctly extracts every bit of m ☺

Summary

NTT

1. We define watermarking PRF against quantum adversaries

   - We define unremovability against adversaries who output quantum program

2. We construct watermarking PRF against quantum adversaries

   - Secret extraction scheme based on LWE
   - Public extraction scheme based on IO

     - Our construction methodology is highly general
       and can be extended to watermarking other primitives such as PKE