Overview of a PQC proposal: the Picnic signature scheme

Luis Brandao and Rene Peralta [1]

[1]National Institute of Standards and Technology (Gaithersburg, USA)

NIST-internal PQC meeting
May 4th, 2018 (Gaithersburg, USA)

Outline

Outline

The Picnic proposal

Highlights:

- A signature scheme (KeyGen, sign, verify)
- No number theoretic or structured hardness assumptions
- Security reduction to symmetric primitives (hash, block-cipher)
- Construction based on a ZKPoK
- Ingredients: Σ protocol, Fiat-Shamir and Unruh transforms, "MPC in the head"

A pre-quantum computers approach

Given P, g, g^x , I claim I know x

A pre-quantum computers approach

Given P, g, g^x , I claim I know x

- I pick random k, c and post g^k, c, and $(k + xc)\ mod\ (P - 1)$.

A pre-quantum computers approach

Given P, g, g^x , I claim I know x

- I pick random k, c and post g^k, c, and $(k + xc)\ mod\ (P-1)$.

- Verification is :

$$g^{k+xc} = g^k (g^x)^c.$$

A pre-quantum computers approach

Given P, g, g^x , I claim I know x

- I pick random k, c and post g^k, c, and $(k + xc)$ mod $(P - 1)$.
- Verification is :

$$g^{k+xc} = g^k (g^x)^c.$$

This is a ZK proof of knowledge (of x) provided you believe I generated **first** g^k and then c at random.

A pre-quantum computers approach

Given P, g, g^x , I claim I know x

- I pick random k, c and post g^k, c, and $(k + xc)$ mod $(P - 1)$.

- Verification is :

$$g^{k+xc} = g^k (g^x)^c.$$

This is a ZK proof of knowledge (of x) provided you believe I generated **first** g^k and then c at random.

- This seems to work: I'll set $c = Hash(g^k)$.

A pre-quantum computers approach

Given P, g, g^x , I claim I know x

- I pick random k, c and post g^k, c, and $(k + xc)$ *mod* $(P - 1)$.
- Verification is :

$$g^{k+xc} = g^k (g^x)^c.$$

This is a ZK proof of knowledge (of x) provided you believe I generated **first** g^k and then c at random.

- This seems to work: I'll set $c = Hash(g^k)$.
- If you want a signature of a message M, set $c = Hash(g^k || M)$.

A pre-quantum computers approach

Given P, g, g^x , I claim I know x

- I pick random k, c and post g^k, c, and $(k + xc)$ mod $(P - 1)$.
- Verification is :

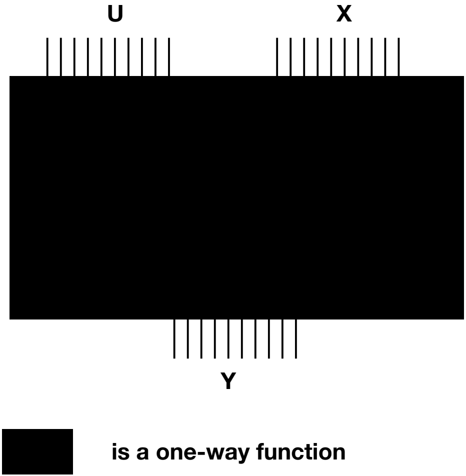
$$g^{k+xc} = g^k (g^x)^c.$$

This is a ZK proof of knowledge (of x) provided you believe I generated **first** g^k and then c at random.

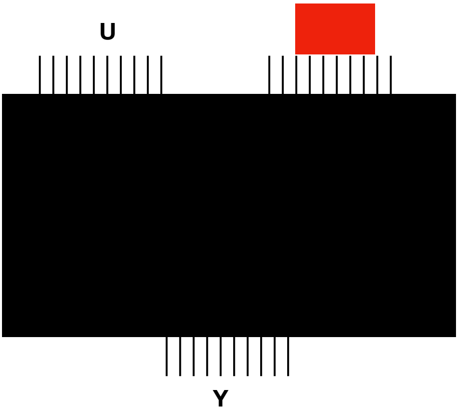
- This seems to work: I'll set $c = Hash(g^k)$.
- If you want a signature of a message M, set $c = Hash(g^k || M)$.

This is a Schnorr signature (I think).

Picnic — illustration at high-level

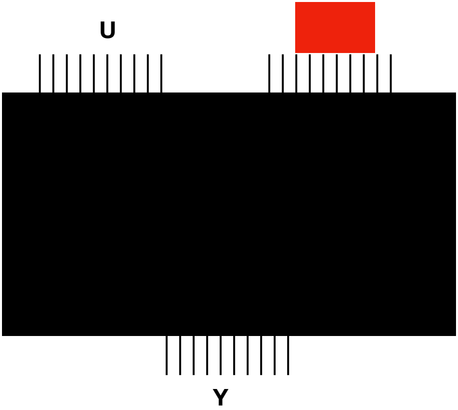


Picnic — illustration at high-level



- given U and Y, I claim I know X

Picnic — illustration at high-level

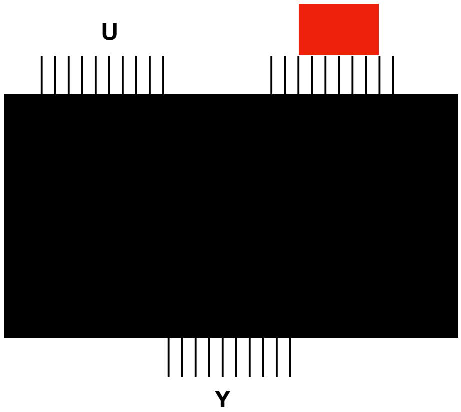


In Picnic :

 **is an encryption function called LOW MC**

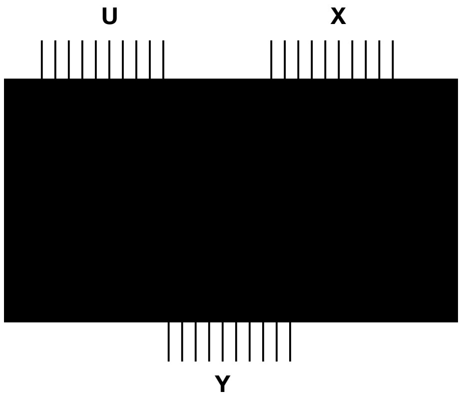
Y is the encryption of U under key X

Picnic — illustration at high-level



- public key pk is (U,Y)
- private key sk is X

Picnic — illustration at high-level



As in Schnorr's signature scheme, we will first need a ZK proof of knowledge of X.

A circuit-based S3PC setup

A circuit-based S3PC setup

ON A CIRCUIT FOR LOW MC

<u>for each input wire:</u> split its boolean value v into three random shares a1, a2, a3 such that v = a1 + a2 + a3

A circuit-based S3PC setup

ON A CIRCUIT FOR LOW MC

<u>for each input wire:</u> split its boolean value v into three random shares a1, a2, a3 such that v = a1 + a2 + a3

Give share i to player Pi

A circuit-based S3PC setup

ON A CIRCUIT FOR LOW MC

<u>for each input wire:</u> split its boolean value v into three random shares a1, a2, a3 such that v = a1 + a2 + a3

Give share i to player Pi

<u>for each gate:</u> propagate the shares from inputs to output

A circuit-based S3PC setup

ON A CIRCUIT FOR LOW MC

<u>for each input wire:</u> split its boolean value v into three random shares a1, a2, a3 such that v = a1 + a2 + a3

Give share i to player Pi

<u>for each gate:</u> propagate the shares from inputs to output

<u>for each output wire:</u> reveal the three shares

A circuit-based S3PC setup

ON A CIRCUIT FOR LOW MC

for each input wire: split its boolean value v into three random shares a1, a2, a3 such that v = a1 + a2 + a3

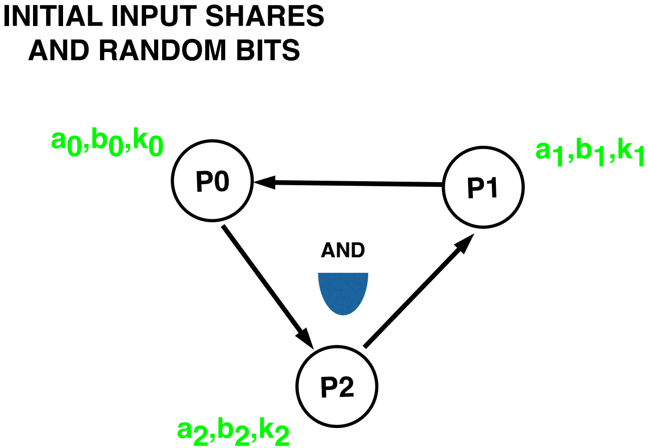
Give share i to player Pi

for each gate: propagate the shares from inputs to output

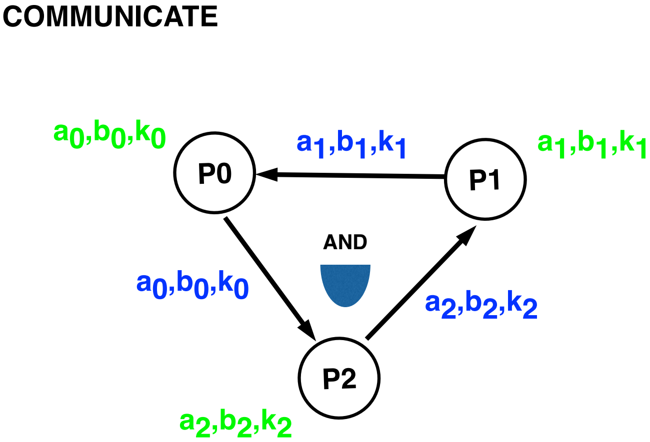
NEXT

for each output wire: reveal the three shares

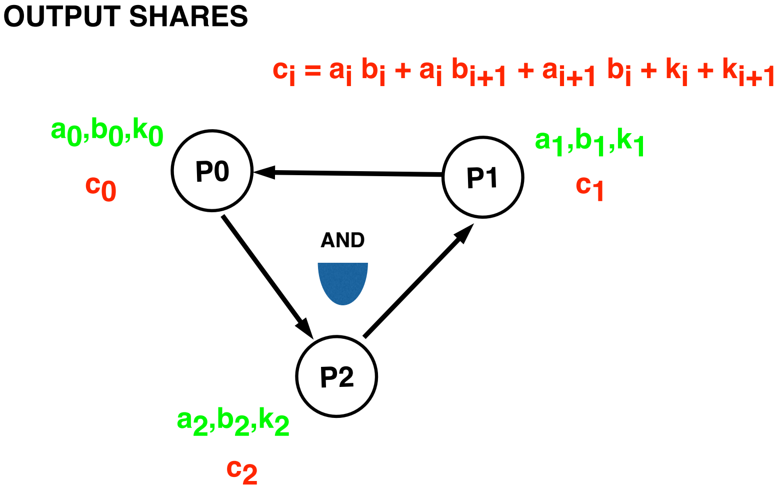
A circuit-based S3PC setup



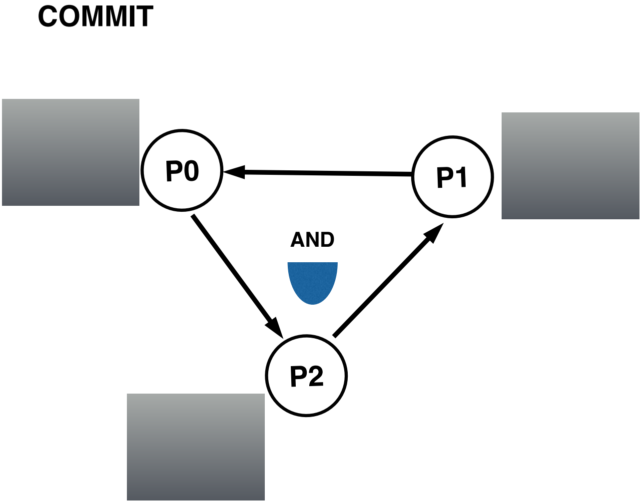
A circuit-based S3PC setup



A circuit-based S3PC setup

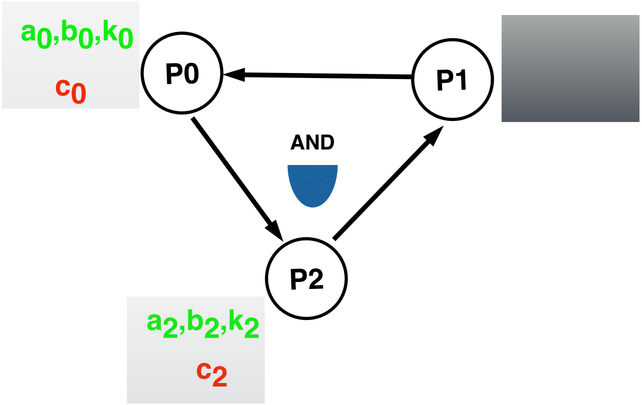


A circuit-based S3PC setup



A circuit-based S3PC setup

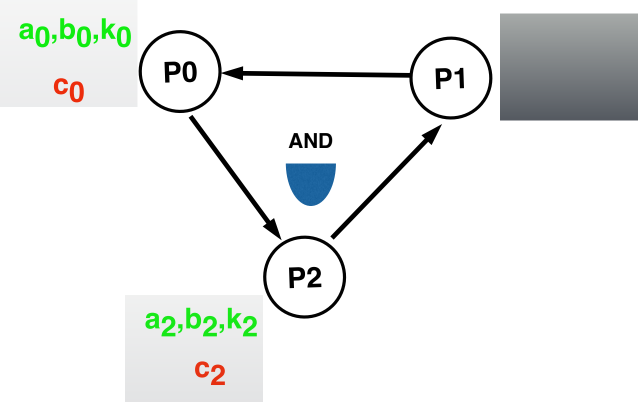
DECOMMIT TWO



A circuit-based S3PC setup

VERIFY

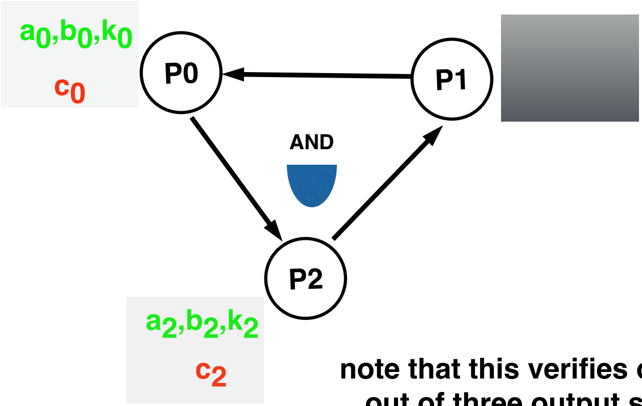
$$c_2 = a_2\, b_2 + a_2\, b_0 + a_0\, b_2 + k_2 + k_0$$



A circuit-based S3PC setup

VERIFY

$$c_2 = a_2 b_2 + a_2 b_0 + a_0 b_2 + k_2 + k_0$$



note that this verifies only one out of three output shares

Outline

Primitives and other ingredients

- Commitment schemes
- Zero Knowledge Proofs of knowledge (ZKPoKs)
- Transformation for non-interactivity: Fiat-Shamir, Unruh
- Low-MC and SHA3
- MPC in the head
- PRNG using SHAKE

Parameters

Parameter Set	S	n	k	s	r	Hash/KDF	Digest length	T	Public key	Private key	Signature
`picnic-L1-FS`	128	128	128	10	20	SHAKE128	256	219	32	16	34000
`picnic-L1-UR`									32	16	53929
`picnic-L3-FS`	192	192	192	10	30	SHAKE256	384	329	48	24	76740
`picnic-L3-UR`									48	24	121813
`picnic-L5-FS`	256	256	256	10	38	SHAKE256	512	438	64	32	132824
`picnic-L5-UR`									64	32	209474