FIPS 202 and Keccak-Derived Functions

John Kelsey, NIST

http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf

http://csrc.nist.gov/groups/ST/toolkit/secure_hashing.html

FIPS 202 is Out!

- We ran a competition for a new *cryptographic hash function* from 2007-2012.
    - 63 submissions from all over the globe.
        - Winner was ***Keccak*** from a Belgian/Italian team.
- After extensive discussions with designers and lots of interaction with community, standardized it
- FIPS 202 out earlier this year.

The Swiss Army Knife of Crypto

- Hash functions are used *everywhere* in crypto, even in places where they're not really appropriate.

- **Digital signatures** (code signing)

- **Message authentication** codes (HMAC)

- **Key derivation** and key agreement schemes (TLS KDF)

- **Proof of knowledge** and timestamping (digital timestamps)

- **Proof of work** and consistency (Bitcoin)

- **Cryptographic pseudorandom bit generation** (HMAC-DRBG)

So, What's in FIPS 202?

- SHA3 functions: fixed-length hash functions just like SHA2
  - **SHA3-224, SHA3-256, SHA3-384, SHA3-512**
  - Number after hyphen is length of output
    - "Drop-in replacements" for SHA2
- Shake functions: extendable-output functions
  - **Shake128, Shake256**
  - Number is security level
- All based on *Keccak* algorithm that won competition

What's Next?

- Keccak has a bunch of nice features that allow new functions to derived from it

- In the pipeline now: (more on the way later)

- **KMAC:** keyed hash construction, like HMAC, but more efficient and with variable-length output.

- **Fast Parallel Hash:** takes advantage of parallelism to get faster, without losing security.

- **TupleHash:** hashes a sequence of strings together in a sensible way.

New Features

- **Domain-separated**: You can't compute KMAC from SHA3 or Shake

- **Customization string**: You can "name" an instance of any function to domain-separate it from all other uses of SHA3-derived functions

- **Variable output length**: specify any output length--get a byte string of whatever length you ask for

KMAC = Keyed Hashing

KMAC128(K,X,96)

- Three security levels: KMAC128, KMAC256, KMAC512
- Variable-length output
- Different output lengths give unrelated outputs
- Domain-separated and customizable

TupleHash = Hashing a List of Strings

TupleHash([s1, s2, s3], 192)

- Always 256-bit security level
- Any or all strings may be empty, but list may not be empty
- Variable length
- Different lengths give unrelated outputs
- Domain-separated and customizable

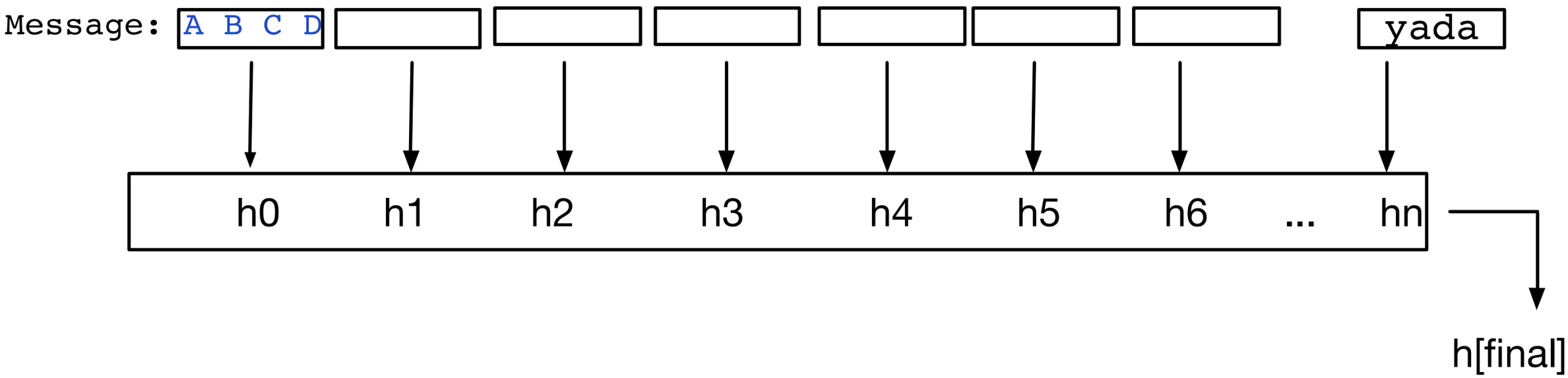
Fast Parallel Hash
Hash big messages in parallel

FPH128(message, blocksize, output_length)

- Two security levels: FPH128 and FPH256
- Variable length output, different output lengths give unrelated outputs
- Domain separated and customizable

There's also a SHA2-derived version which works very differently, and has different properties. I'm not talking about it here.

Fast Parallel Hash



- Break long message into blocks
  - Blocksize in bytes, between 2^8 and 2^40.
- Hash each block and store result
- Finally hash all the results together sequentially

Wrapup

- Keccak has a bunch of nice features built in and provided by designers.
- We plan to make use of them in our Keccak-derived standards
  - **KMAC**
  - **TupleHash**
  - **FPH**
- All with **customization strings** and **variable-length output**
- We hope to have the first of these out for public comment soon!

Bonus Slides for Questions

Domain Separated

- This means you can't compute KMAC by calling SHAKE or SHA3 or TupleHash or FPH
  - NOTE: Very different from how HMAC works!
- Outputs of different functions ***completely unrelated***
- This should make it a little harder to shoot yourself in the foot with these functions

Customization Strings

This is domain separation controlled by the user

- KMAC["KDF"](K,X)

is *completely unrelated to*

- KMAC["Message Block"](K',X')

Like strong typing for uses of a hash function

Customization Strings

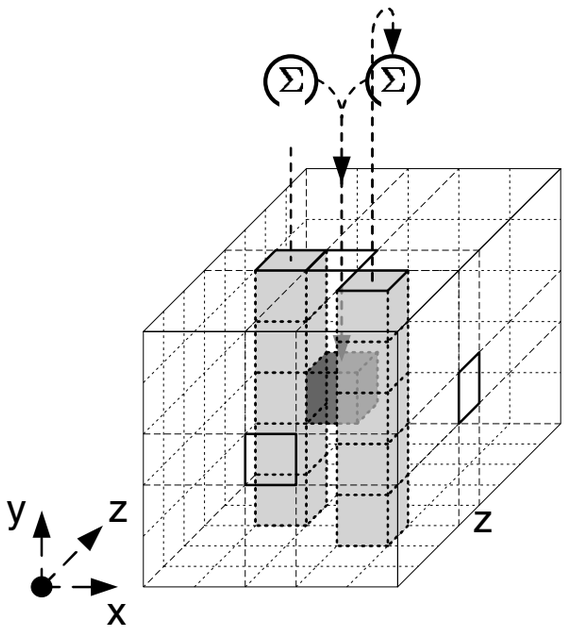
- All new Keccak-derived functions will support "customization strings" to let users further domain-separate them

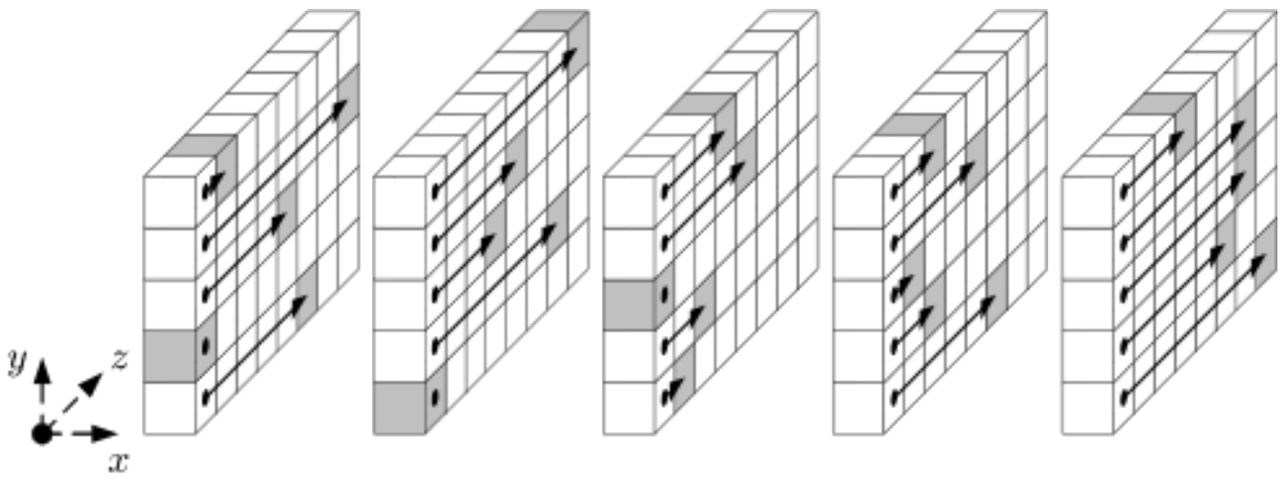
- We plan to introduce a new SP to allow customization strings for Shakes and SHA3, too

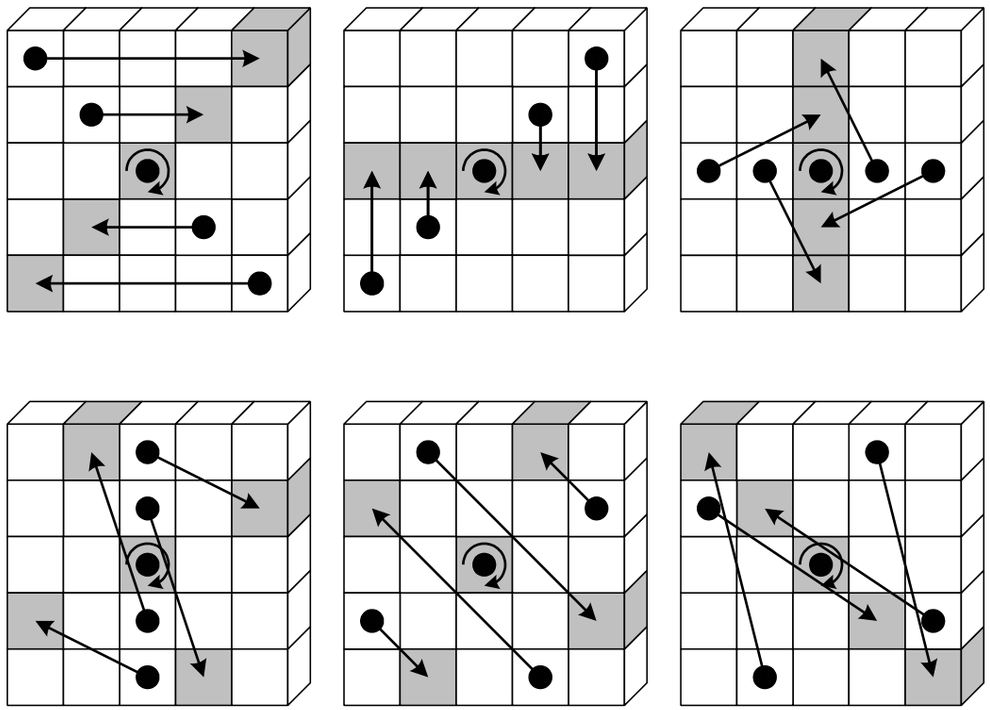
Variable Length

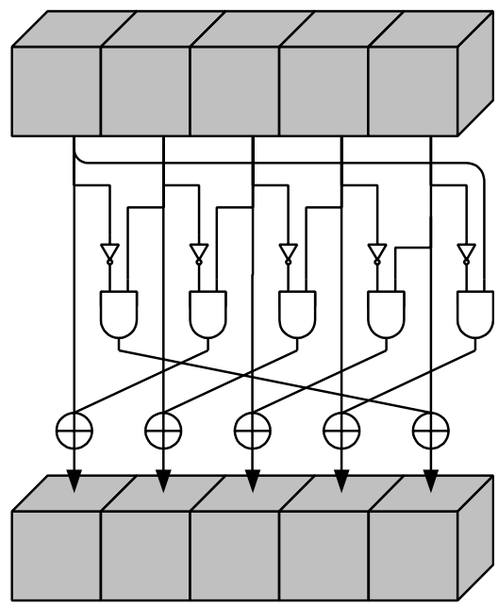
- The Shakes are variable length, but different output lengths give related outputs.
- Shake128(X,96) = **ABC**
- Shake128(X,128) = **ABC**D
- KMAC, TupleHash, and FPH are variable length, and different output lengths give unrelated outputs
- KMAC128(K,X,96) = **EFG**
- KMAC128(K,X,128) = **HIJ**K
- This is easier to use--harder for a designer to shoot himself in the foot

Keccak looks nothing like MD4



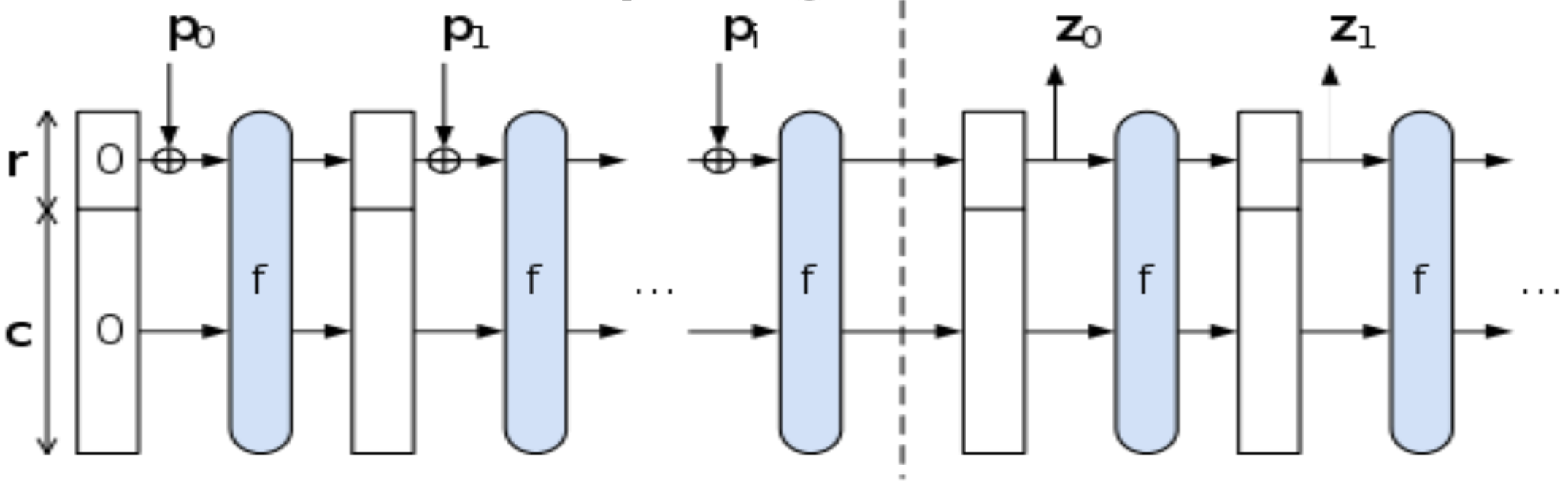






Images from Keccak submission

No More MD: Keccak is a Sponge



- Security based on fixed-length permutation f
- r bits of message XORed into state at a time
- c bits left untouched
- Can generate arbitrary number of output bits

Hash Function = "digital fingerprint" of a message

- **Collision-resistance**: It should be very hard to find X,Y so that

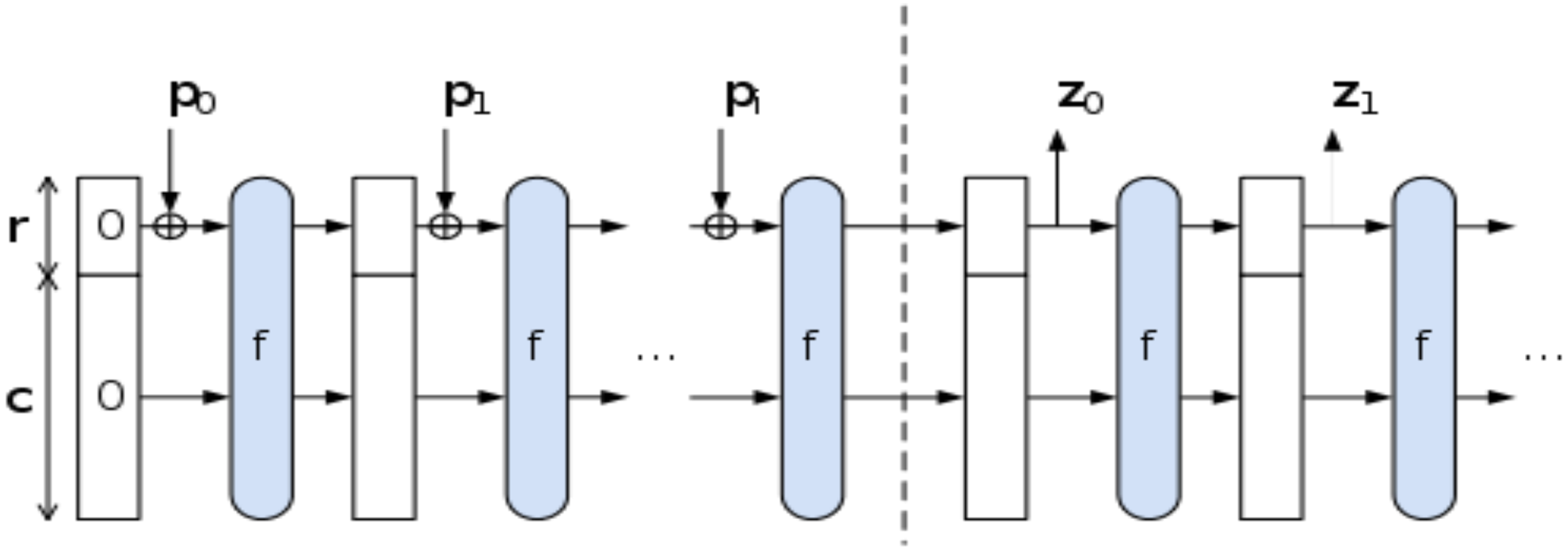
$$\text{hash}(X) = \text{hash}(Y)$$

- **Preimage-resistance**: Given some target T, it should be even harder to find an X so that

$$\text{hash}(X) = T$$

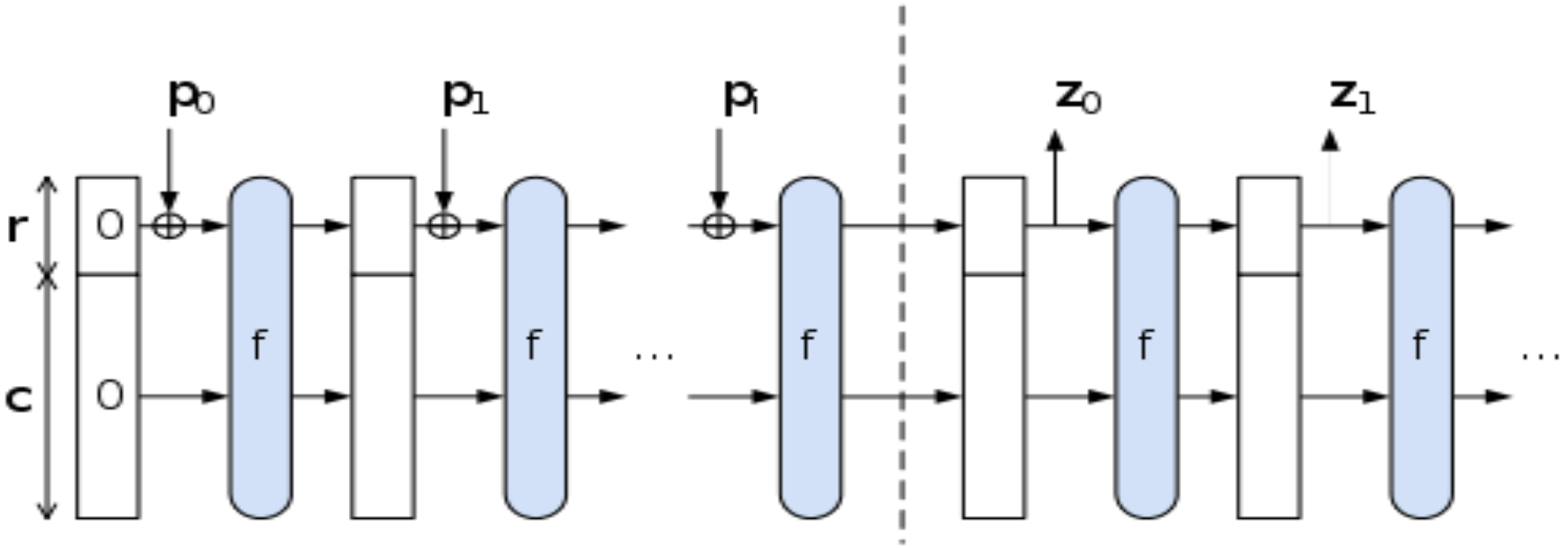
- **Pseudorandomness**: If you don't know all the bits of X, hash(X) should look very random to you.

Capacity and Security



- A sponge has collision and preimage resistance of c/2 bits.
- Finding a collision or preimage is equally hard

Security/Performance Tradeoff



- Bigger c → smaller r → slower hashing
- The choice of c is a tunable parameter in Keccak
  - Allows a security/performance tradeoff
  - Security level is c/2.