Amortized Complexity of Zero-Knowledge Proofs Revisited: Achieving Linear Soundness Slack

Ronald Cramer (CWI)

Ivan Damgård (AU)

Chaoping Xing (NTU)

ChenYuan (NTU)

Eprint 2016/681

Integer One-Way Function (iOWF)

- maps integers to finite group G
- hard to invert
- additively homomorphic

f: Z -> G (in paper: integer *vectors* to G)

f(x+y) = f(x)+f(y)

Integer One-Way Function (iOWF)

- maps integers to finite group G
- hard to invert
- additively homomorphic

f: Z -> G (in paper: integer *vectors* to G)

f(x+y) = f(x)+f(y)

Examples:

- encryption functions for many lattice-based crypto-systems
- lattice based hash-functions
- integer commitment schemes

Zero-Knowledge for iOWFs

Prover P claims he knows a small(short) preimage x for output value y = f(x).

Zero-Knowledge for iOWFs

Prover P claims he knows a small(short) preimage x for output value y = f(x).

Useful in many contexts:

Zero-Knowledge for iOWFs

Prover P claims he knows a small(short) preimage x for output value y = f(x).

Useful in many contexts:

- Prove that ciphertext is well-formed, so it decrypts uniquely

Zero-Knowledge for iOWFs

Prover P claims he knows a small(short) preimage x for output value y = f(x).

Useful in many contexts:

- Prove that ciphertext is well-formed, so it decrypts uniquely
- Preimage of hash function is short enough, so collisions are hard to find

Simplistic Zero-Knowlegde

x **P** y=f(x) **V** **claim: |x|< b**

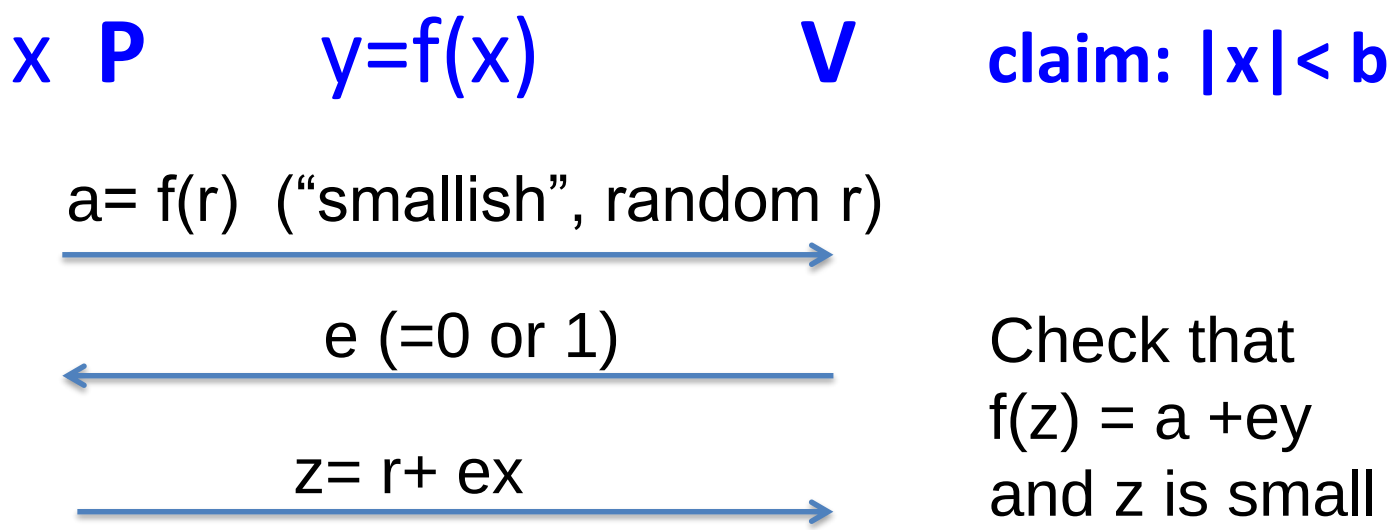
a= f(r) (“smallish”, random r)

e (=0 or 1)

z= r+ ex

Check that
f(z) = a +ey
and z is small

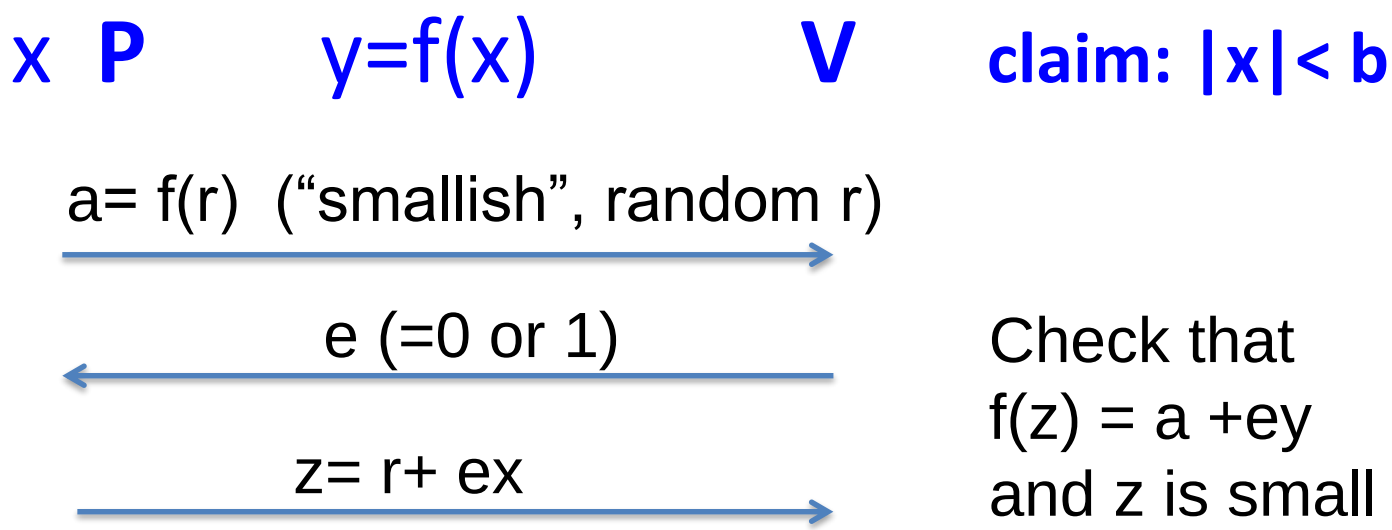
Simplistic Zero-Knowlegde



Problems (1): to make this be ZK, need that |r| is bigger than b by exponentially large factor, in security parameter k.

Then, preimage we can extract from cheating prover is also large: we say the **soundness slack** is exp(k)

Simplistic Zero-Knowlegde



Problems (2): must repeat protocol k times to get exp(-k) error probability. Taking e from larger domain does not work.
We say the **overhead** is k.

State of the Art and Our Results

Prove knowledge of a single preimage: we do not know how to reduce both overhead and soundness slack.
Consider instead images y1,…,yn and the *amortized* cost of proving preimage knowledge.

State of the Art and Our Results

Prove knowledge of a single preimage: we do not know how to reduce both overhead and soundness slack.
Consider instead images y1,…,yn and the *amortized* cost of proving preimage knowledge.

[CD09]: overhead O(1), soundness slack exp(k)
[BDLN16]: overhead O(1), soundness slack $O(n\, k^{\log(k)})$

State of the Art and Our Results

Prove knowledge of a single preimage: we do not know how to reduce both overhead and soundness slack.
Consider instead images y1,…,yn and the *amortized* cost of proving preimage knowledge.

[CD09]: overhead O(1), soundness slack exp(k)
[BDLN16]: overhead O(1), soundness slack $O(n\, k^{\log(k)})$

This work:

overhead O(1), soundness slack O(k)

State of the Art and Our Results

Prove knowledge of a single preimage: we do not know how to reduce both overhead and soundness slack.
Consider instead images y1,…,yn and the *amortized* cost of proving preimage knowledge.

[CD09]: overhead O(1), soundness slack exp(k)
[BDLN16]: overhead O(1), soundness slack $O(n\, k^{\log(k)})$

This work:

overhead O(1), soundness slack O(k)

- Need that n is k^2 constants are small, practical solution.
- Can reduce to $k^{1.5}$ (and better in subsequent work) theoretical interest.

The Construction

”Imperfect Proof” borrowed from [BDLN16]:

Cut-and-choose + Lyubashevsky’s rejection sampling.

Overhead O(1), Soundness slack O(1)

Ensures that we can extract from P a small preimage of all but k of the yi.

Improved version in [dPL17].

The Construction

”Imperfect Proof” borrowed from [BDLN16]:

Cut-and-choose + Lyubashevsky’s rejection sampling.

Overhead O(1), Soundness slack O(1)

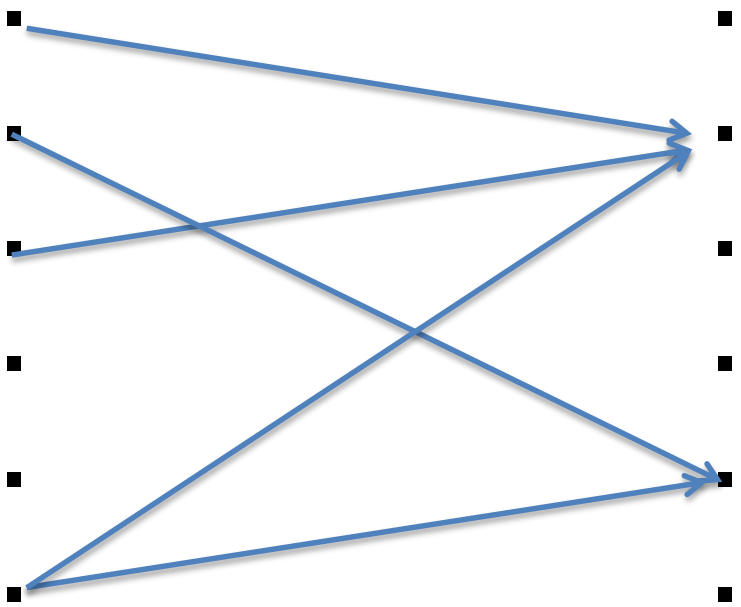
Ensures that we can extract from P a small preimage of all but k of the yi.

Improved version in [dPL17].

Main Protocol (our contribution)

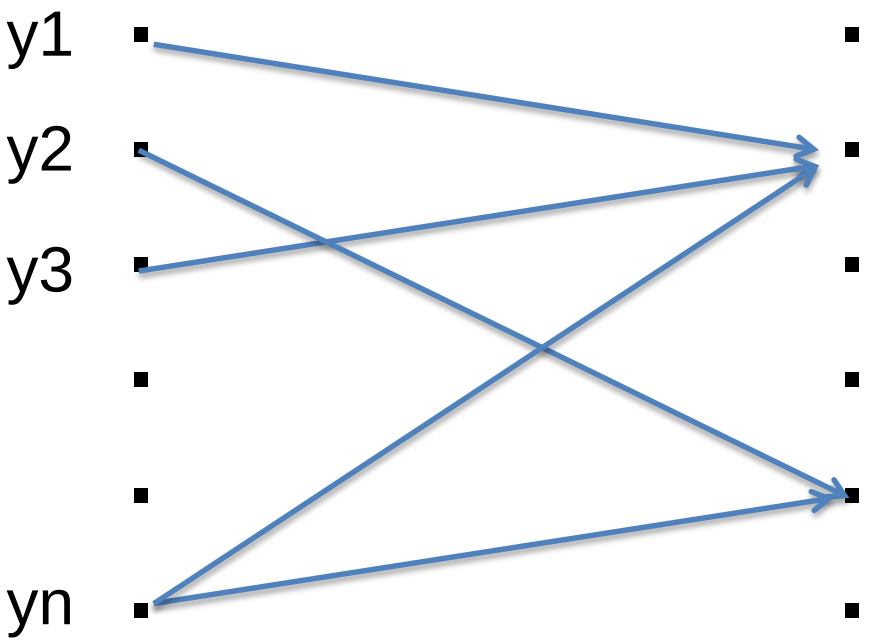
Use Imperfect proof, homomorphic property and a bipartite graph with good expansion properties to get protocol from which we can extract all preimages.

Using a Bipartite graph



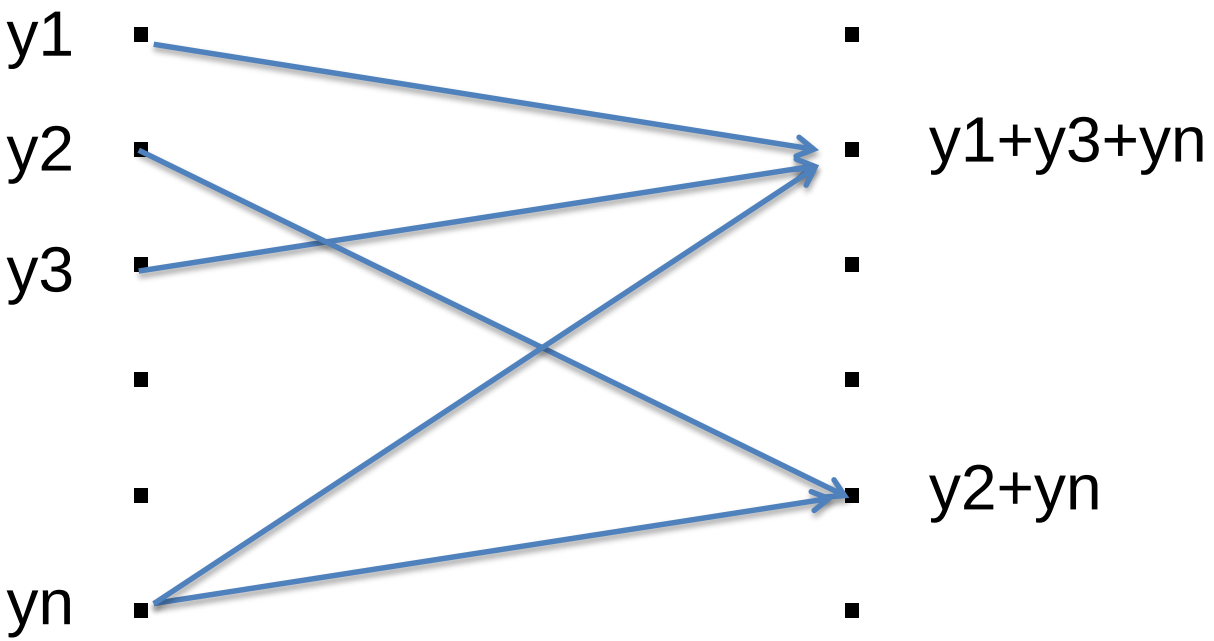
- n nodes on the left and right

Using a Bipartite graph



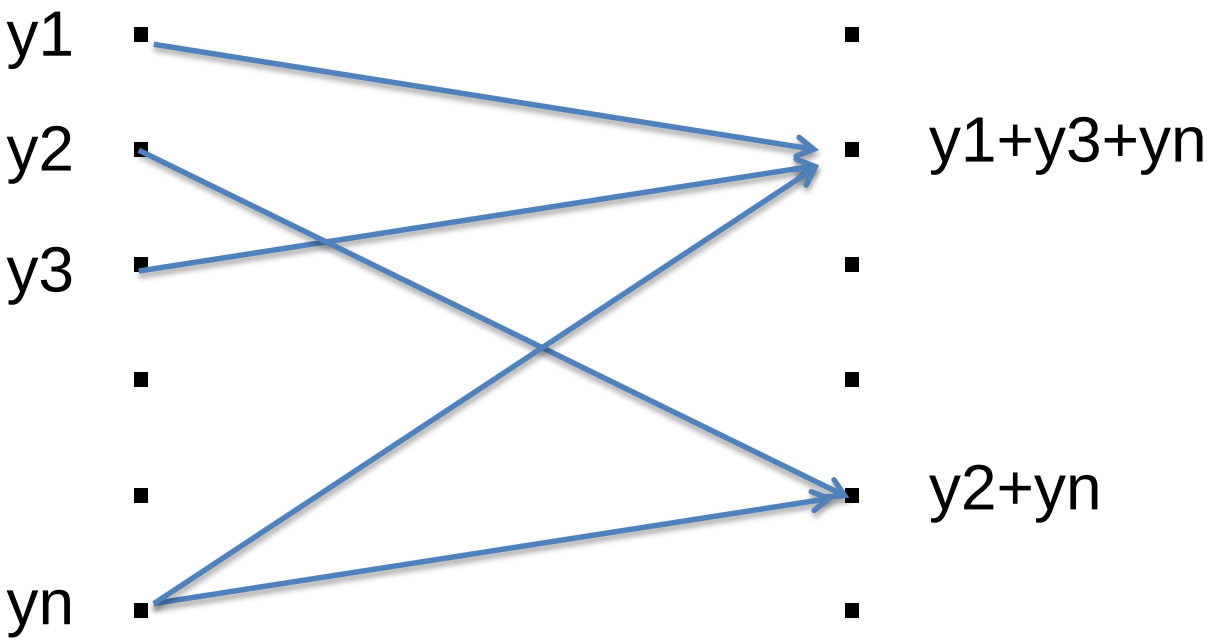
- n nodes on the left and right
- Assign yi to i'th node on the left.

Using a Bipartite graph



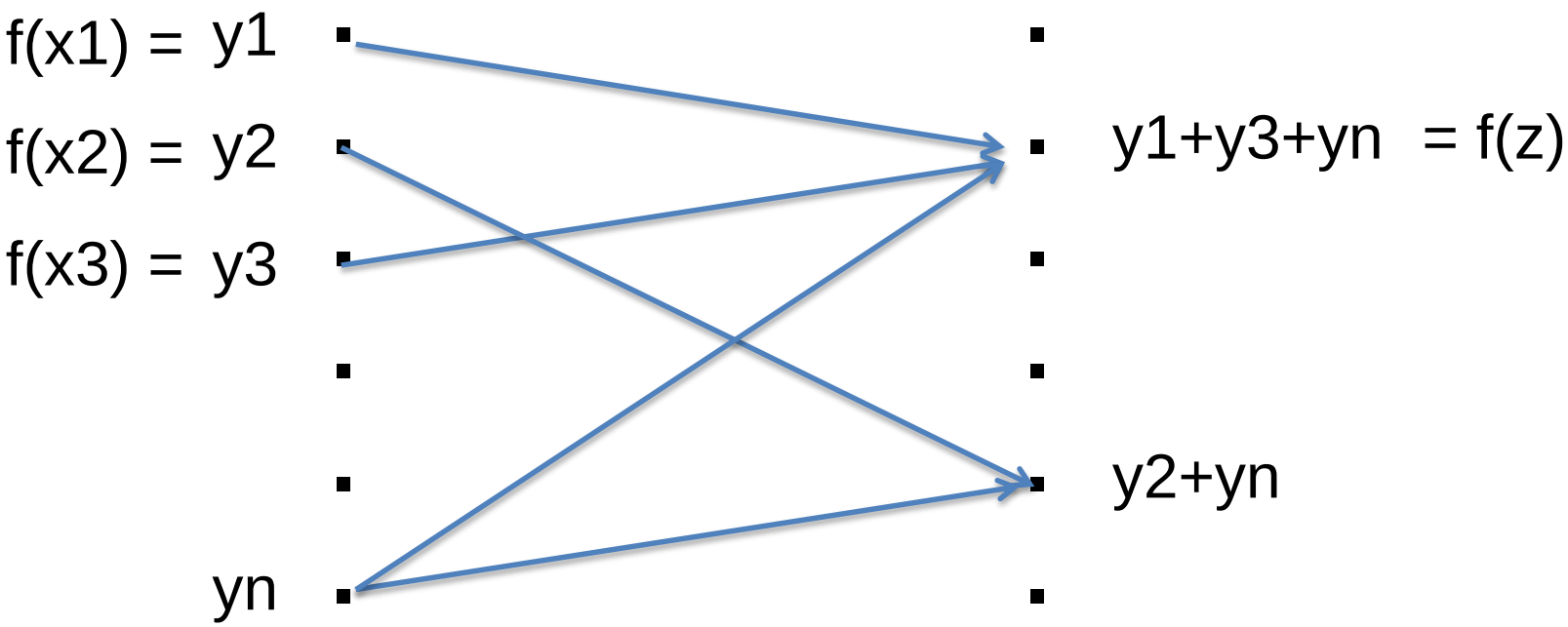
- n nodes on the left and right
- Assign yi to i'th node on the left.
- Assign to each node on the right the sum of values from it neighbors

Using a Bipartite graph 2



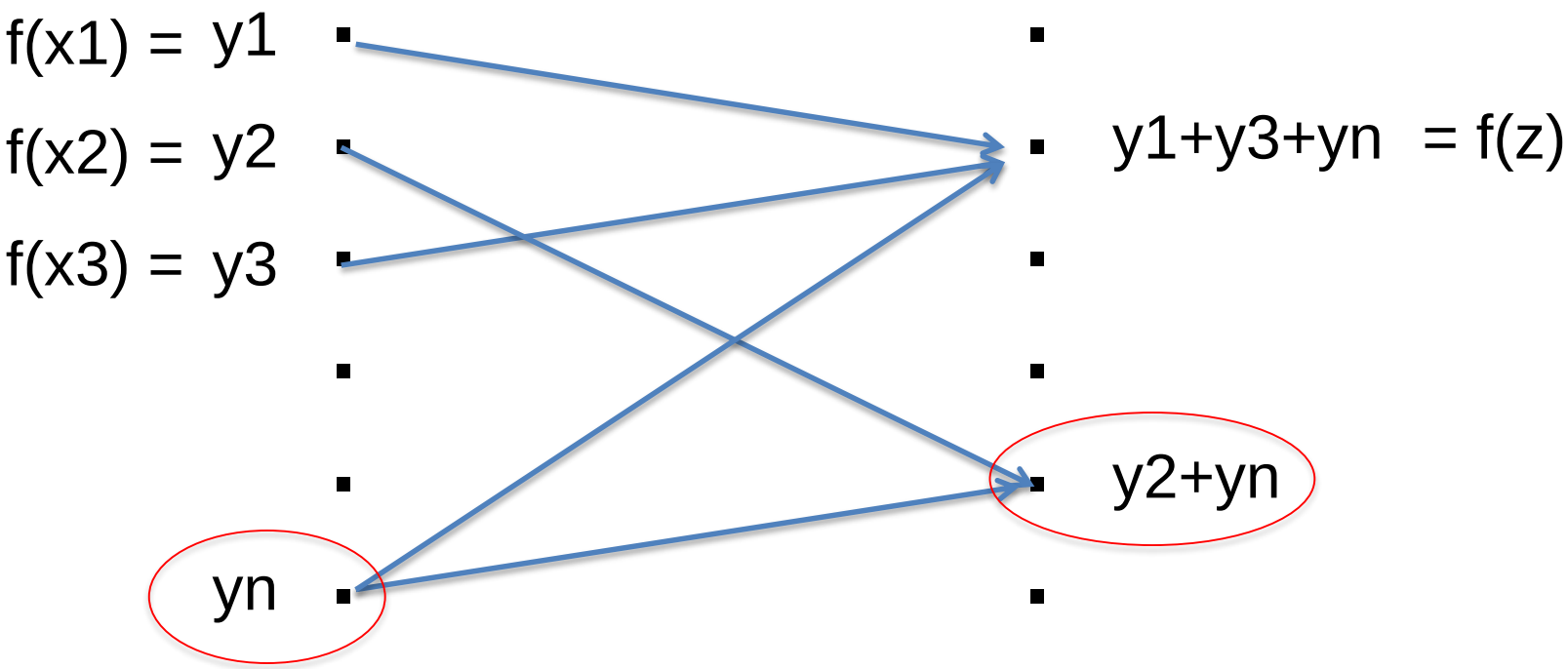
- Use Imperfect Proof on values on the left, and also on values on the right.

Using a Bipartite graph 2



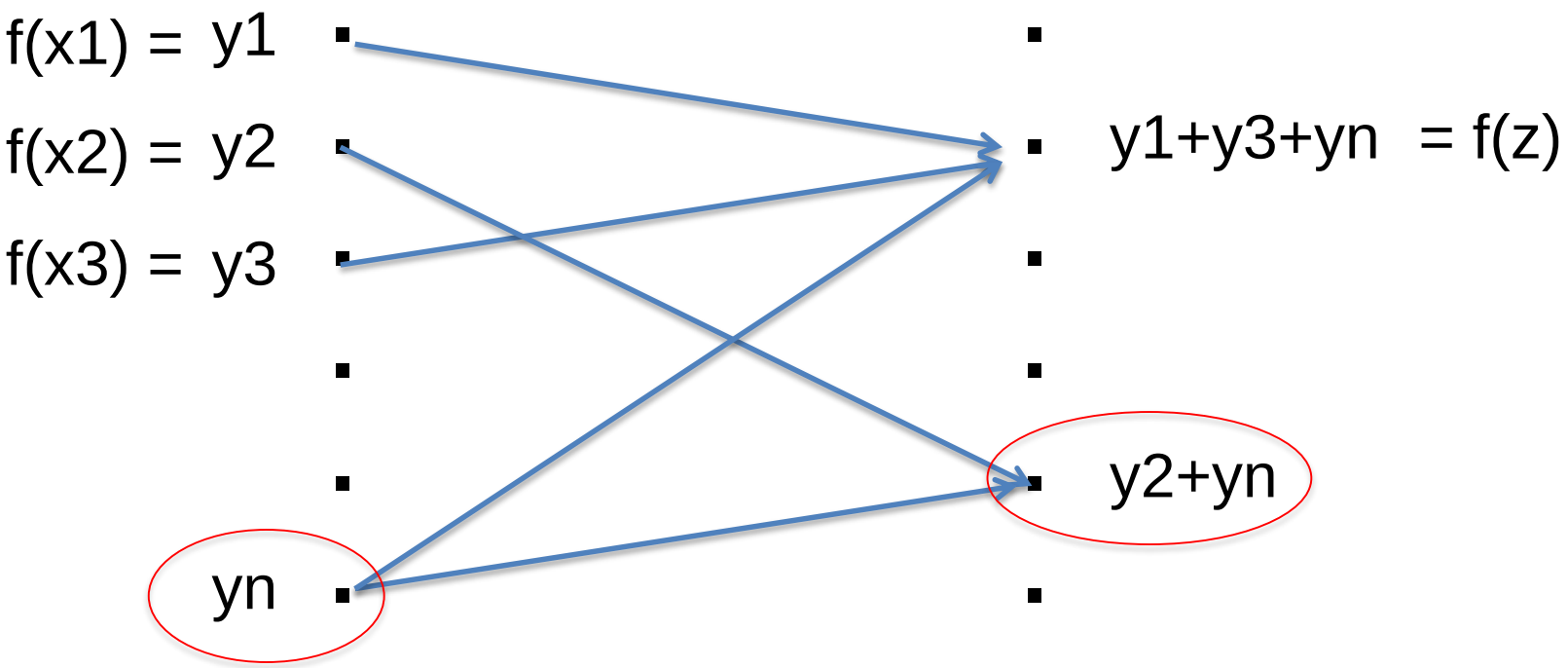
- Use Imperfect Proof on values on the left, and also on values on the right.
- We can extract from P small preimages of almost all instances.

Using a Bipartite graph 2



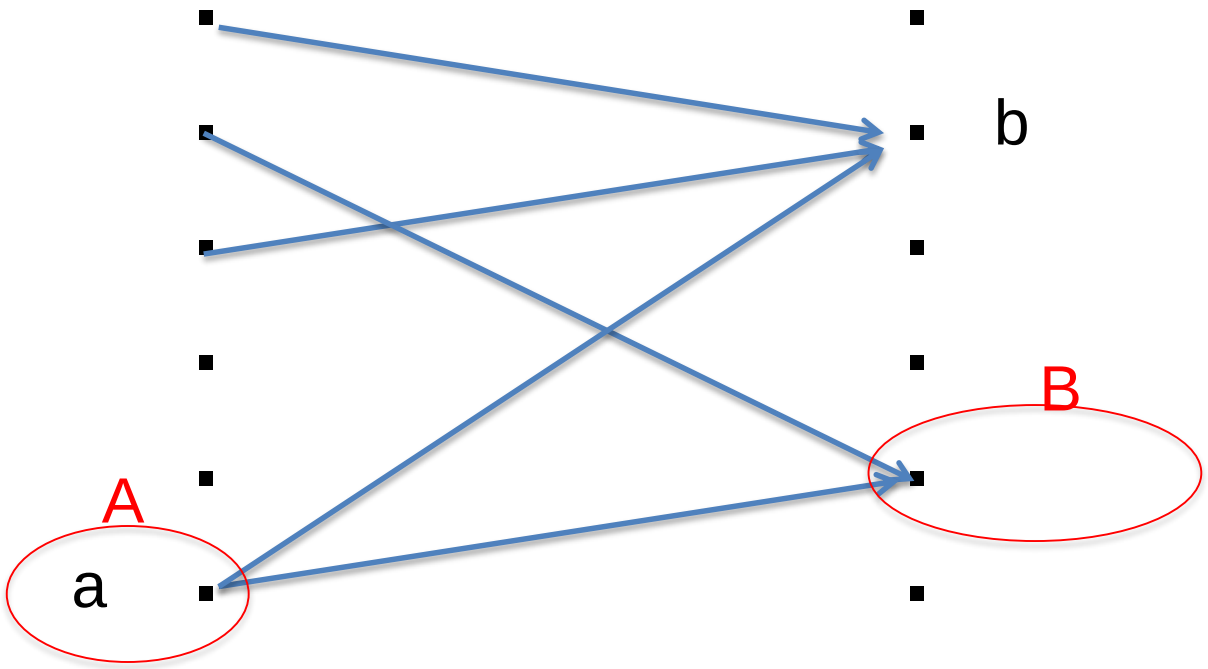
- Use Imperfect Proof on values on the left, and also on values on the right.
- We can extract from P small preimages of almost all instances.
- Say we fail on 1 instance on both sides

Using a Bipartite graph 3



- We failed on yn, but if we can find a place on the right where 1) we succeeded and 2) yn is “alone”, we are good:
- yn= f(z)- y1- y3 = f(z- x1- x3)
- If |z|, |x1|, |x3| are < b, then |z-x1-x3| < 3b

Requirements on the graph



- In-degree on the right: O(k) - then soundness slack is O(k).
- Strong unique neighbor property: Consider any two subsets of size k, A on the left, B on the right. For each a in A there exists b not in B such that {a} = A ∩ Neighborhood(b) - then extraction works.

Construction of good graphs 1

In general, related to graphs with good expansion properties, but known results don’t do what we want. We get the result we need from universal hash functions.

Construction of good graphs 1

In general, related to graphs with good expansion properties, but known results don't do what we want. We get the result we need from universal hash functions.

Let $p > 2k+1$ be a prime and F the field with p elements. A member in our family H is defined by h in F. We define $h(a_0,a_1) = ha_0+a_1$.

Construction of good graphs 1

In general, related to graphs with good expansion properties, but known results don't do what we want. We get the result we need from universal hash functions.

Let $p > 2k+1$ be a prime and F the field with p elements. A member in our family H is defined by h in F. We define $h(a_0,a_1) = ha_0+a_1$.

Set of nodes on the left: X= FxF

Set of nodes on the right: Y= HxF

Construction of good graphs 1

In general, related to graphs with good expansion properties, but known results don't do what we want. We get the result we need from universal hash functions.

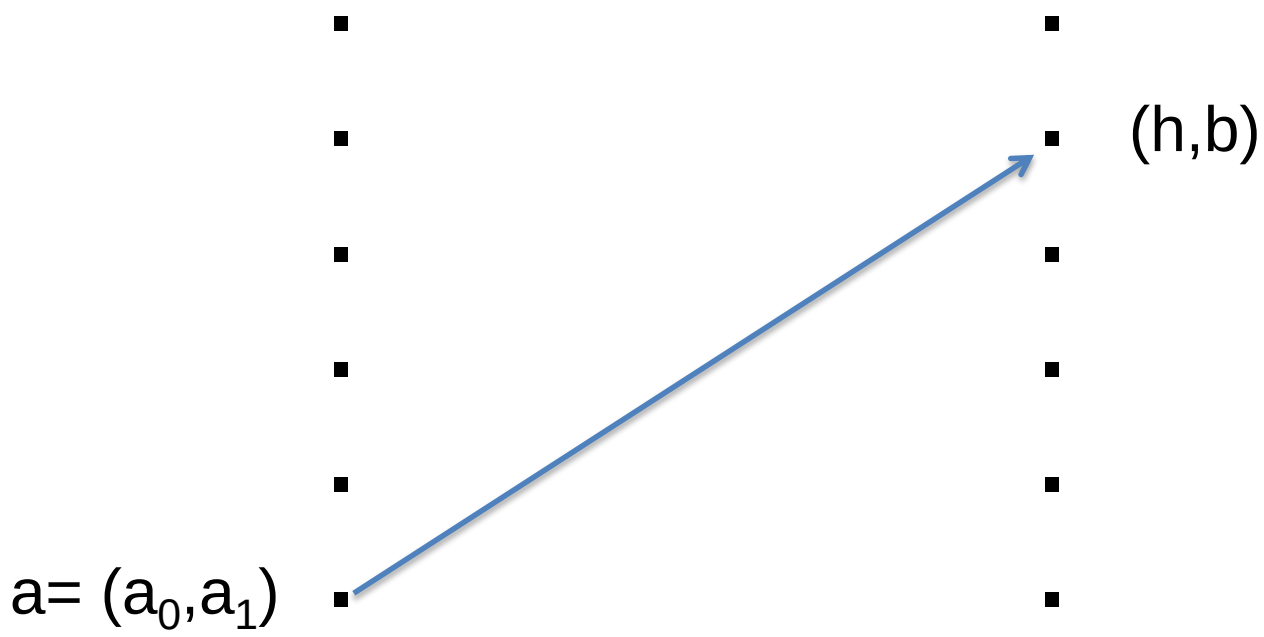
Let $p > 2k+1$ be a prime and F the field with p elements. A member in our family H is defined by h in F. We define $h(a_0,a_1) = ha_0+a_1$.

Set of nodes on the left: X= FxF

Set of nodes on the right: Y= HxF

Edge from (a_0,a_1) to (h,b) iff $h(a_0,a_1) = b$.

Construction of good graphs 2



- Edge exists iff $h(a_0,a_1) = ha_0+a_1 = b$
- We get a good graph with $n <= 16k^2$ nodes on each side and in-degree O(k).
- In-degree is clear, for strong unique neighbor property, see paper.

Alternative Construction

..can be based on certain known graphs with good expansion properties.

We adapt previous proofs techniques to get the properties we need.

We get $n = O(k^3)$ and strong unique neighbor property only holds in a probabilistic sense.

BUT: is still useful even when $n << k^3$: implies a protocol that reduces the number of unknown preimages significantly.

Can combine with first result to get soundness slack O(k), overhead O(1) with $n= O(k^{1.5})$.

Acknowledgement: to Omer Reingold for an idea leading to the $n=O(k^2)$ result.

Acknowledgement: to Omer Reingold for an idea leading to the $n=O(k^2)$ result.

Thanks!