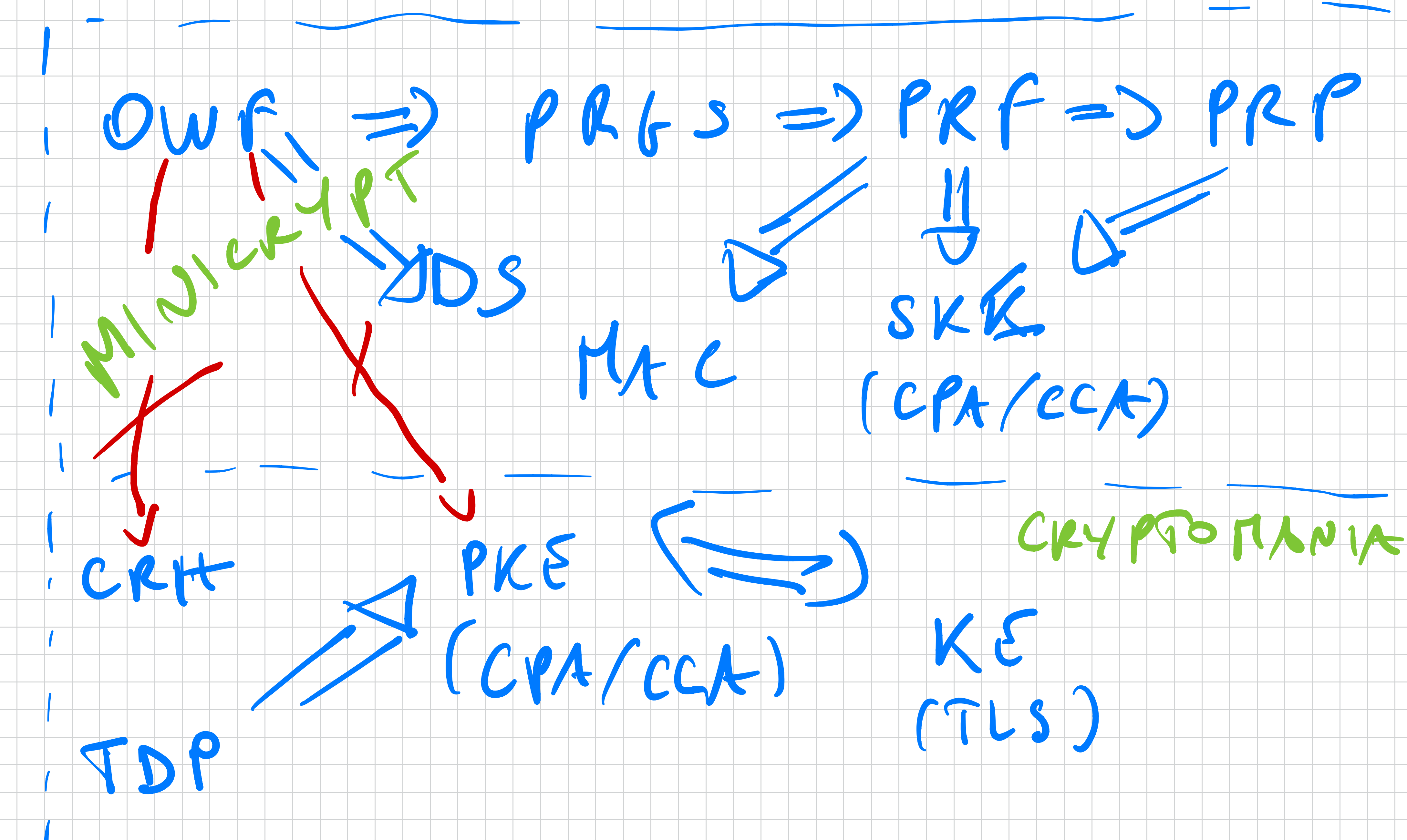
CRYPTOMANIA

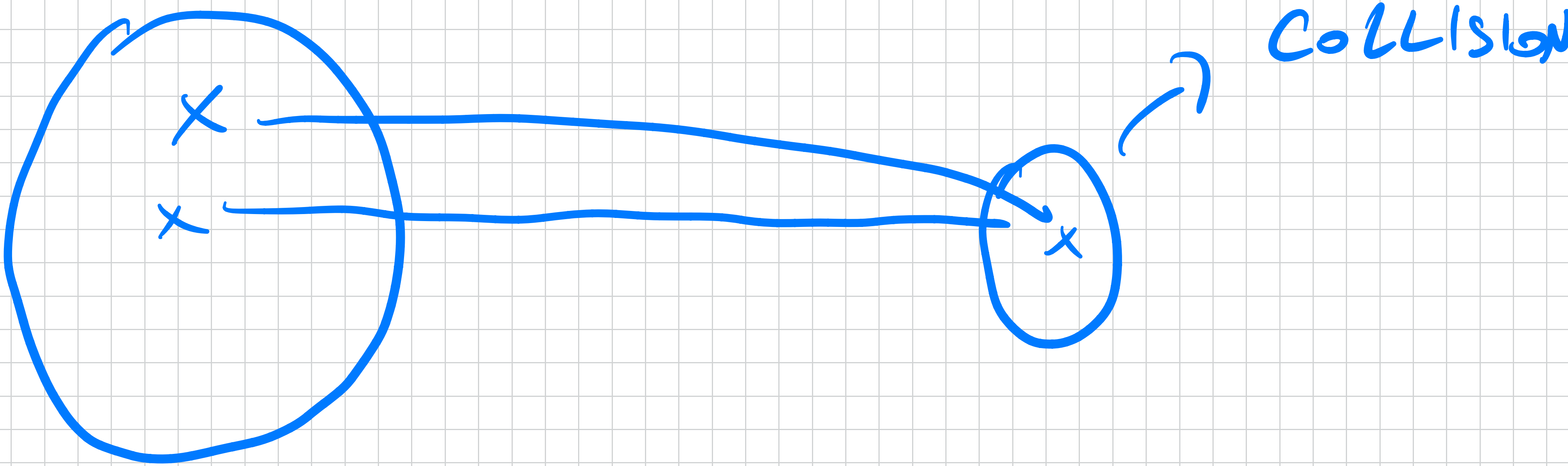


COLLISION-RESISTANT HASH

This about families of functions

$$\mathcal{H} = \{ h_s : \{0,1\}^{\ell} \to \{0,1\}^{n} \}_{s \in \{0,1\}^{\lambda}}$$

s.t. $\ell = \ell(n) >> n$.



Recall: When we studied PRFs we have seen the construction $F(H)$ which is a way to extend the domain of ANY PRF F.

When the seed is PUBLIC, CRH are only possible for comp. bounded attackers.

Many real-world examples: MD5, SHA1, SHA2, SHA3, Merkle trees, ...

DEF We say that $\mathcal{H}$ is collision resistant if $\forall$ PPT $\mathcal{A}$:

$$\Pr\left[\mathrm{GAME}^{crh}_{\mathcal{H},\mathcal{A}}(\lambda) = 1\right] \leq \mathrm{negl}(\lambda).$$

<u>$\mathrm{GAME}^{crh}_{\mathcal{H},\mathcal{A}}(\lambda)$</u>

$\mathcal{A}$ $\qquad\qquad\qquad\qquad$ $\mathcal{C}$

$\xleftarrow{\quad s \quad}$ $\qquad$ $s \leftarrow U_\lambda$

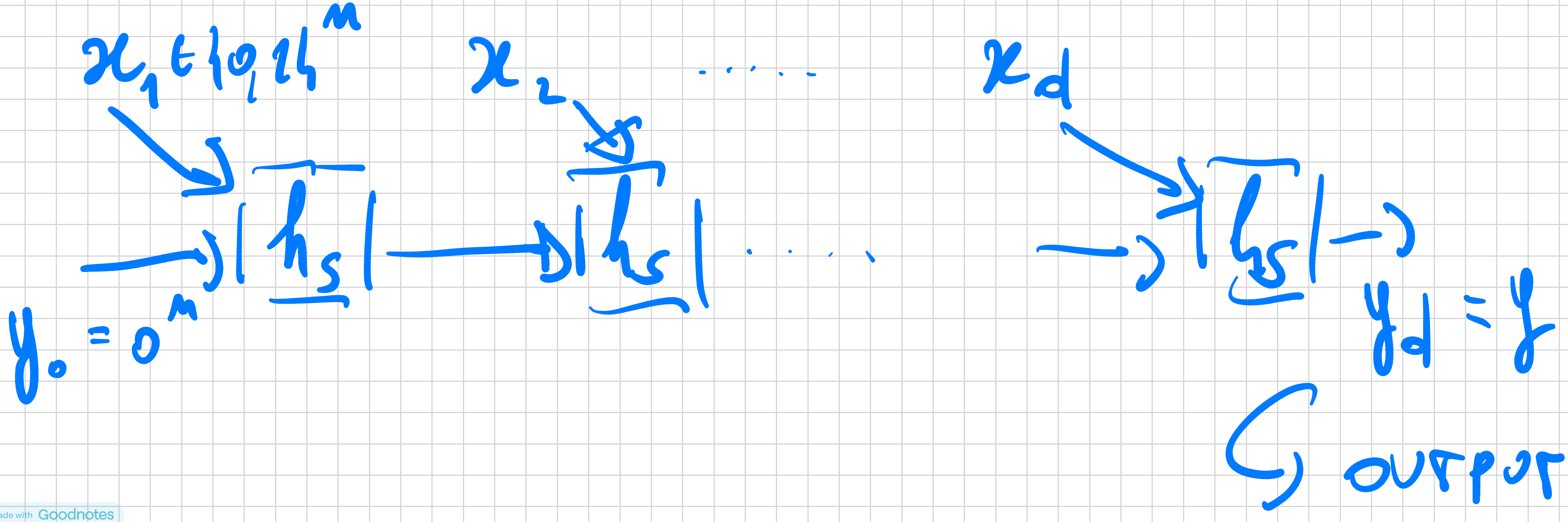
$\xrightarrow{\quad x, x' \quad}$
$(x \neq x')$

Output 1 iff $h_s(x) = h_s(x')$

Typical application: Hash a long msg and then Sign / MAC / PRF it.
An important remark: Why do we need a seed? In fact, SHA for instance does not have any seed!
We can't rule out the attacker $A_{x,x'}$ which has a collision x, x' hard-wired and outputs it.

First construction: Merkle-Damgaard Transform. This is behind MD5, SHA1, SHA2. The idea is to obtain CRH H with domain $\{0,1\}^*$ assuming CRH

$$h_s : \{0,1\}^{2n} \to \{0,1\}^n.$$

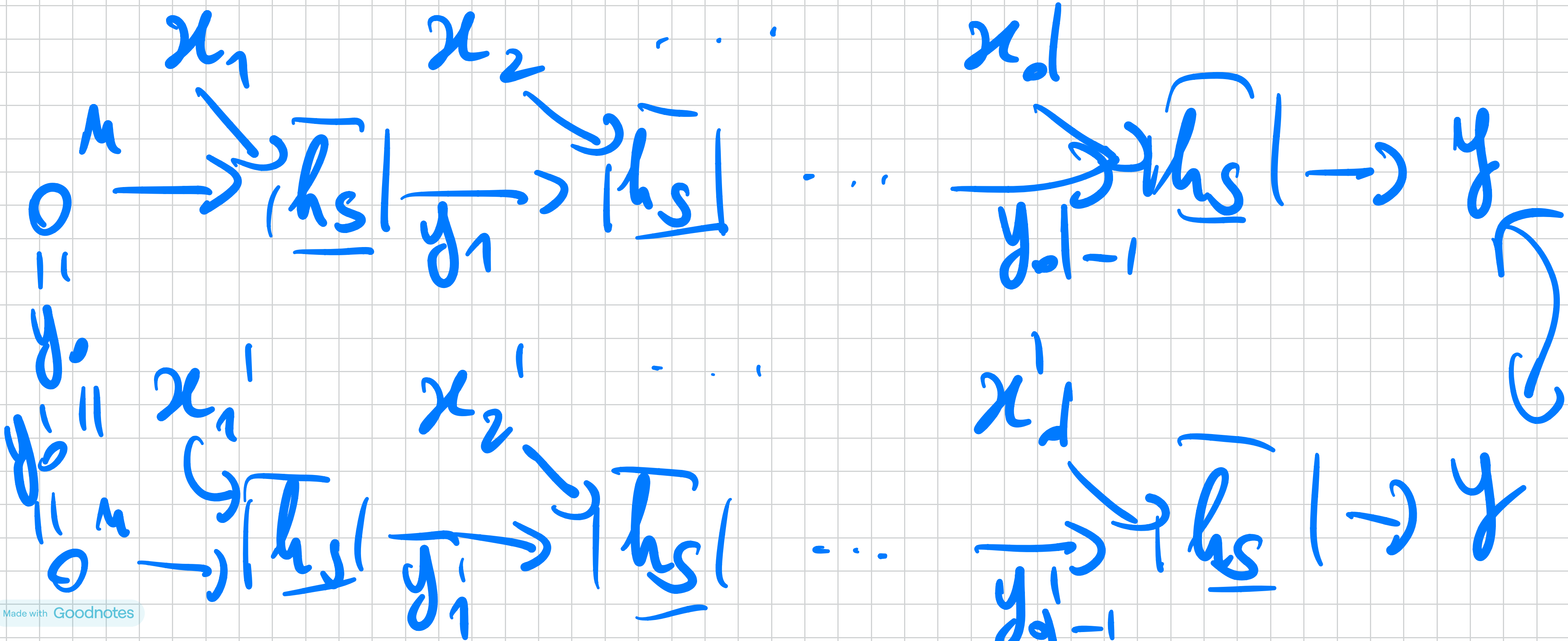


THM Assuming $\{h_s : \{0,1\}^{2n} \to \{0,1\}^n\}$ is CR, Then $\{H_s : \{0,1\}^{d \cdot n} \to \{0,1\}^n\}$ is also CR for every fixed $d \in \mathbb{N}$.

Proof. We basically observe that a collision $x, x' \in \{0,1\}^{n \cdot d}$ $(x \neq x')$ for H_s implies a collision for h_s. Moreover the latter is efficiently computable. Thus immediately implies a reduction. Thus, no attacker can find collisions for H_s.

Let $x = (x_1, \ldots, x_d) \neq x' = (x'_1, \ldots, x'_d)$

s.t. $H_s(x) = H_s(x') = y$



Looking backwards (from right to left), let $i \in [d]$ be the largest index s.t. $h_s(x_i, y_{i-1}) = h_s(x_i', y_{i-1}')$ and $(x_i, y_{i-1}) \neq (x_i', y_{i-1}')$. Such i always exist because $x \neq x'$. Now, (x_i, y_i) and (x_i', y_i') are a collision for h_s. $\blacksquare$

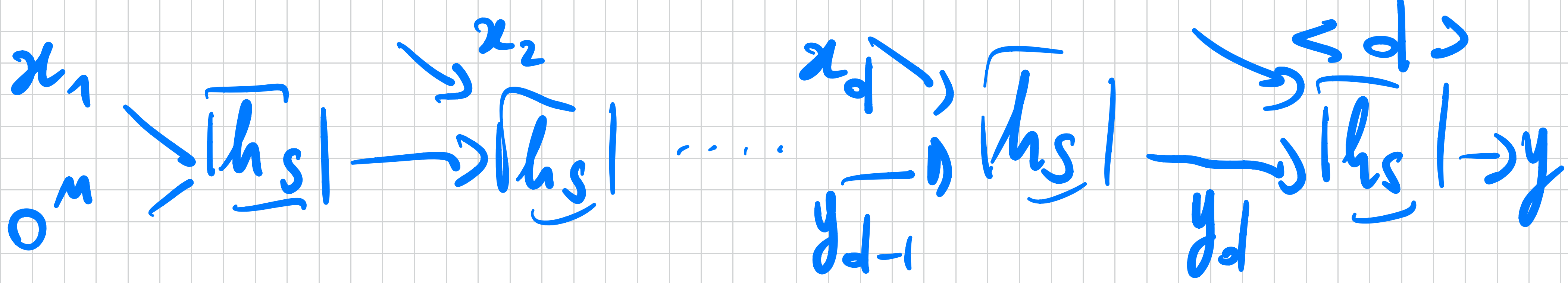
Unfortunately, this does not work for $\{0,1\}^*$. This is because we can't rule out that $h_s(0^{2n}) = 0^n$ while $\{h_s\}$ is still CR. If this is true, then for any x:

$$H_s(x) = H_s(0^n \| x)$$

$$= H_s(0^{2n} \| x)$$

$$\vdots$$

To avoid it: Encode x s.t. no legal encoded x can be a suffix of another input.



$\langle d \rangle$: # of blocks encoded using m bits.

Note: You can only hash inputs of at most 2^m blocks, but this is HUGE for real values of m (e.g. $m = 256$).

THM Assuming $\{h_s\}$ as before, Then the modified $\{H_s\}$ as above is CR for $\{0,1\}^*$.

Proof. We follow the same strategy. Let $x = (x_1, \ldots, x_d)$ and $x' = (x'_1, \ldots, x'_{d'})$ be a collision for H_s.

There are 2 cases:

- $d \neq d'$. Then $(\langle d \rangle, y_d) \neq$

$(\langle d' \rangle, y'_{d'})$ but $h_s(\langle d \rangle, y_d)$
$= h_s(\langle d' \rangle, y'_{d'})$.

- $d = d'$. As before. $\boxtimes$

How to build $h_s : \{0,1\}^{2n} \to \{0,1\}^n$?
In practice: heuristically (MD5, SHA1, SHA2). In theory: Either you use number-theoretic assumptions (FACTORING, DISCRETE LOG, POST-QUANTUM assumptions).

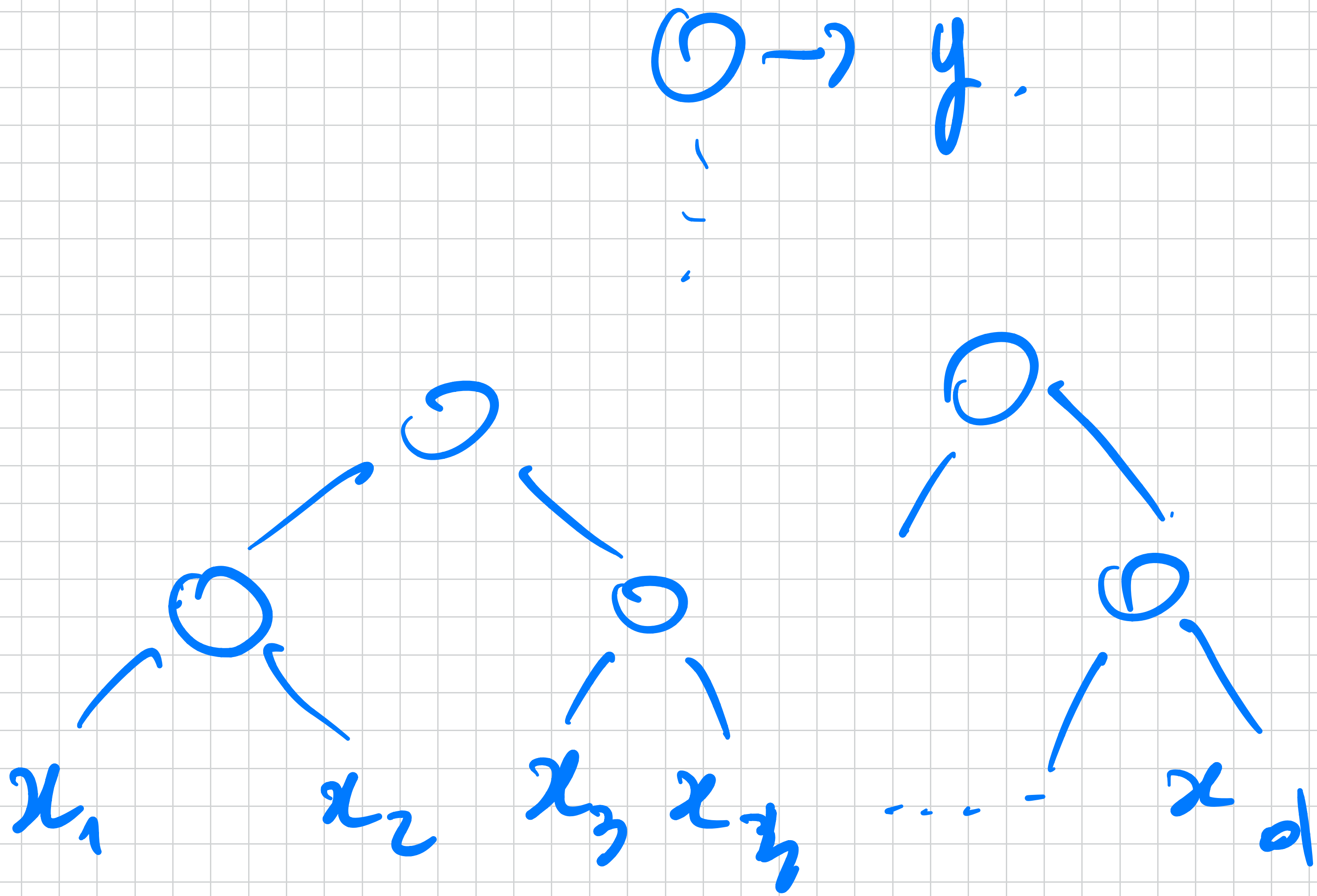
Plausible practical solution: Use AES.

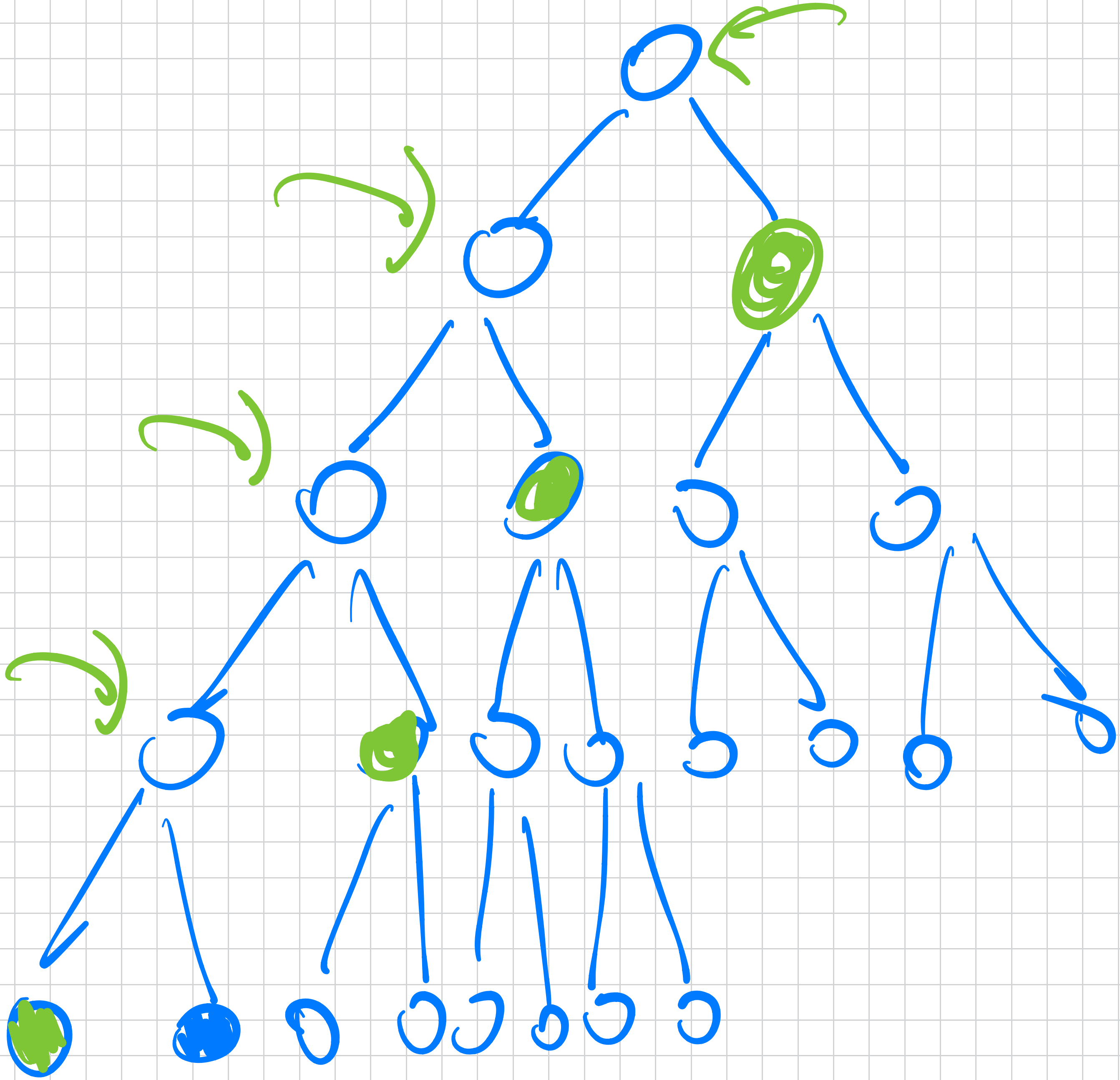
$$h_s(x_1, x_2) = AES(x_1, x_2) \oplus x_2$$

Caveat: We can prove it secure only assuming AES is an IDEAL CIPHER: a TRULY RANDOM PERMUTATION for every choice of the key.

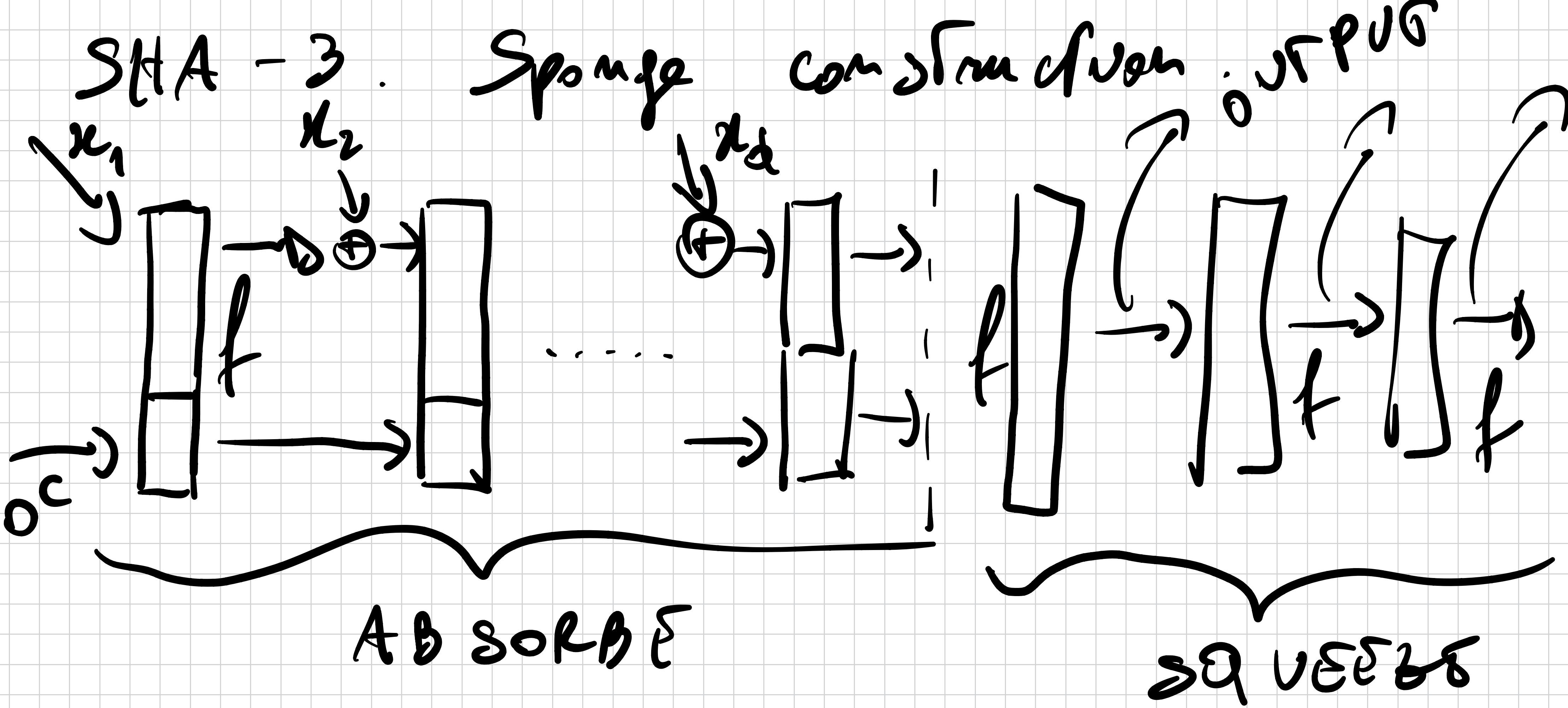
A couple of more facts about hash functions:

1) Alternative constructions.





SHA-3 . Sponge construction



f : PUBLIC RANDOM PERMUTATION.

Another application of hash functions is to build MACs: the standard HMAC. Based on the idea:

$$\mathrm{Tag}(K, m) = H(K \| m).$$

If $H(\cdot)$ is a RANDOM FUNCTION (RANDOM ORACLE MODEL) this is ok. Not secure if $H(\cdot)$ built from MERKLE DAMGAARD.

Simple exercise: length extension attack

Given $\tau = H_s(K \| m)$ we can forge on $m^* = (m_1 \| \cdots \| m_d \| m')$ by outputting $\tau^* = h_s(\tau \| m')$

$$= H_s(K \| m^*)$$

It can be adapted to the case with

The SUFFIX-FREE encoding.

$$HMAC((K_1, K_2), m) = h_S(K_2 \| H_S(K_1 \| m))$$

K_1, K_2 derived from some K.

NUMBER THEORY

We will introduce some concrete assumptions: FACTORING, DISCRETE LOG, LEARNING WITH ERRORS.

Number Theory is about modular arithmetic, i.e. mod n, namely

$$\mathbb{Z}_n = \{0, 1, 2, \ldots, n-1\}.$$

Then you can have structures like

$(\mathbb{Z}_n, +)$, $(\mathbb{Z}_n, +, \cdot)$

$+, \cdot$ are mod n.

For instance $(\mathbb{Z}_n, +)$ is a GROUP.
The situation is different for $(\mathbb{Z}_n, \cdot)$, it is not always a group.

LEMMA If $\gcd(a, n) > 1$, then $a \in \mathbb{Z}_n$ not invertible mod n w.r.t. "$\cdot$".