Programmable Hash Functions and their applications

Dennis Hofheinz, Eike Kiltz

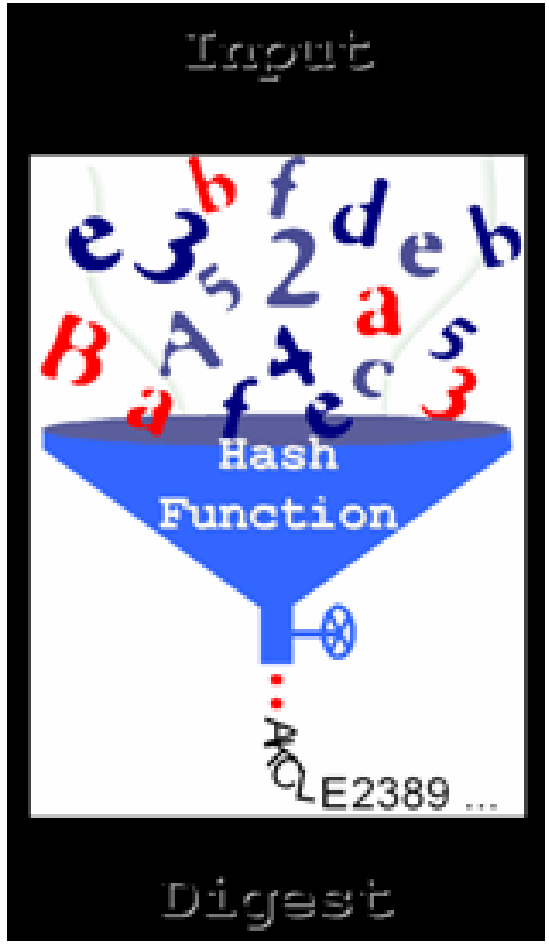
CWI, Amsterdam

Overview

1. Hash functions
2. Programmable hash functions
3. Applications of PHF's
4. Instantiations of PHF's

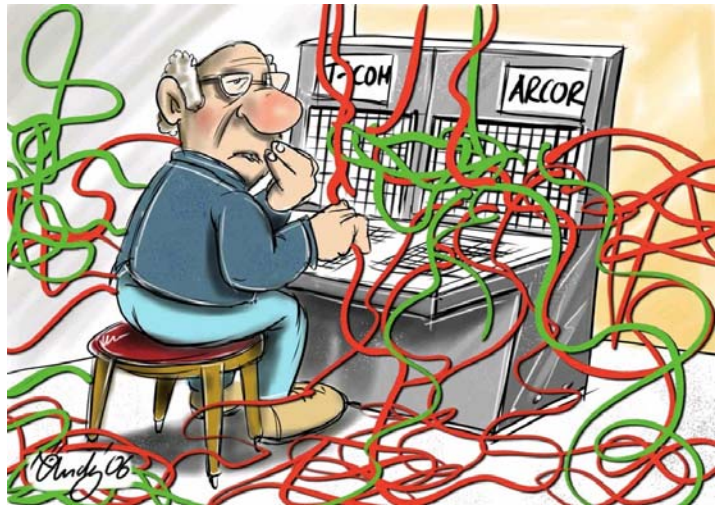
Hash Functions

- **Cryptographic hash function**
  $H_\kappa : \{0,1\}^* \rightarrow \{0,1\}^n$
  "keyed"
- **Applications**
  - Message Integrity
  - Password verification
  - Building block for cryptographic protocols
  - …



What is a “good” hash function?

- **Message integrity**
  - Collision resistant: finding $x \neq x'$ such that $H_k(x)=H_k(x')$ infeasible
- **Password verification**
  - One-way/preimage resistant: given z, finding x: $H_k(x)=z$ infeasible
- **Building block for cryptographic protocols**
  - ???
  - depends on protocol



One of most important algorithms

- **Signature example: RSA "full-domain hash"**
  - Public-key: $N=pq$, $e \in Z^*_{(p-1)(q-1)}$
  - Secret-key: $d=e^{-1} \bmod (p-1)(q-1)$
  - $H: \{0,1\}^* \rightarrow Z_N$ hash function
  - **Sign(sk,m):** $\sigma = H(m)^d \bmod N$
  - **Verify(pk,sig):** $\sigma^e = H(m) \bmod N$?

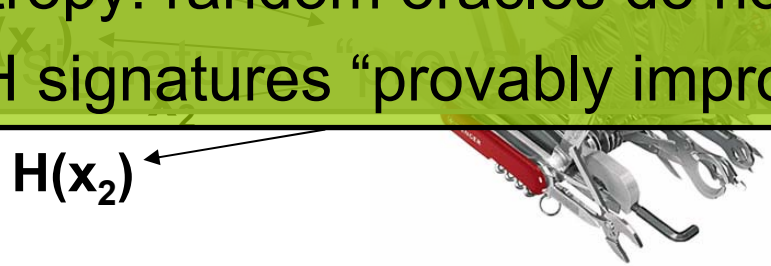
- **Necessary conditions for hash H:**
  - one-way!
  - collision resistant!
  - pseudorandom generator/function?
  - enough !? $\Rightarrow$ "provable security"

Provable security

- We want it all: random oracles
  - **H idealized!** Every new evaluation of H yields uniform bit-string
  - Information theoretic tool for analyzing protocols

However

- Too much entropy: random oracles do not exist
- Last talk: FDH signatures "provably improvable"



- **Theorem:**
  - If RSA is one-way and H is a random oracle then the RSA-FDH signature scheme is "secure"

Goal of this work

- **Security notion for hash functions**
  - Programmable hash functions
    - information-theoretic tool for analyzing protocols
    - mimics programmability property of RO
  - sufficiently strong to be useful
  - sufficiently weak to be (efficiently) achievable

but

Overview

1. Hash functions
2. Programmable hash functions
3. Applications of PHF's
4. Instantiations of PHF's

Throughout this talk

- **G = cyclic group**
- **g = generator of G**
- **Discrete logarithm is hard in G**
  - Given u find x such that $g^x=u$ is hard
- **Sometimes pairing**
  - e: GxG $\rightarrow G_T$, $e(g^x,g^y)=e(g^y,g^x)=e(g,g)^{xy}$

(m,n)-Programmable Hash Function

- **PHF = (Kg, Eval, TrapGen,TrapEval)**
  - $Kg() \rightarrow \kappa$
  - $Eval(\kappa,X) \rightarrow H_\kappa(X) \in G$
  - $TrapKg(g,u) \rightarrow (\kappa,t)$
  - $TrapEval(t,X) \rightarrow (a(X),b(X))$ s.t. $H_\kappa(X) = g^{a(X)}\, u^{b(X)}$
    - "make H dependent on g and u

"programmability"

- **For all strings $X^{(1)}, \ldots, X^{(m)} \neq Y^{(1)}, \ldots, Y^{(n)}$:**

$$\Pr\left[\begin{array}{l} b(X^{(i)}) = 0,\ i=1\ldots m \text{ and} \\ b(Y^{(j)}) \neq 0, j=1\ldots n \end{array}\right] = \text{non-negl.}$$

  - implies:

    Red $X^{(i)}$: $H_\kappa(X^{(i)}) = g^{a(X)}$

    Green $Y^{(i)}$: $H_\kappa(Y^{(i)}) = g^{a(Y)}\, u^{b(Y)}$

Toy instantiation

H = random oracle $\Rightarrow$ H = (poly,1)-PHF

Because:

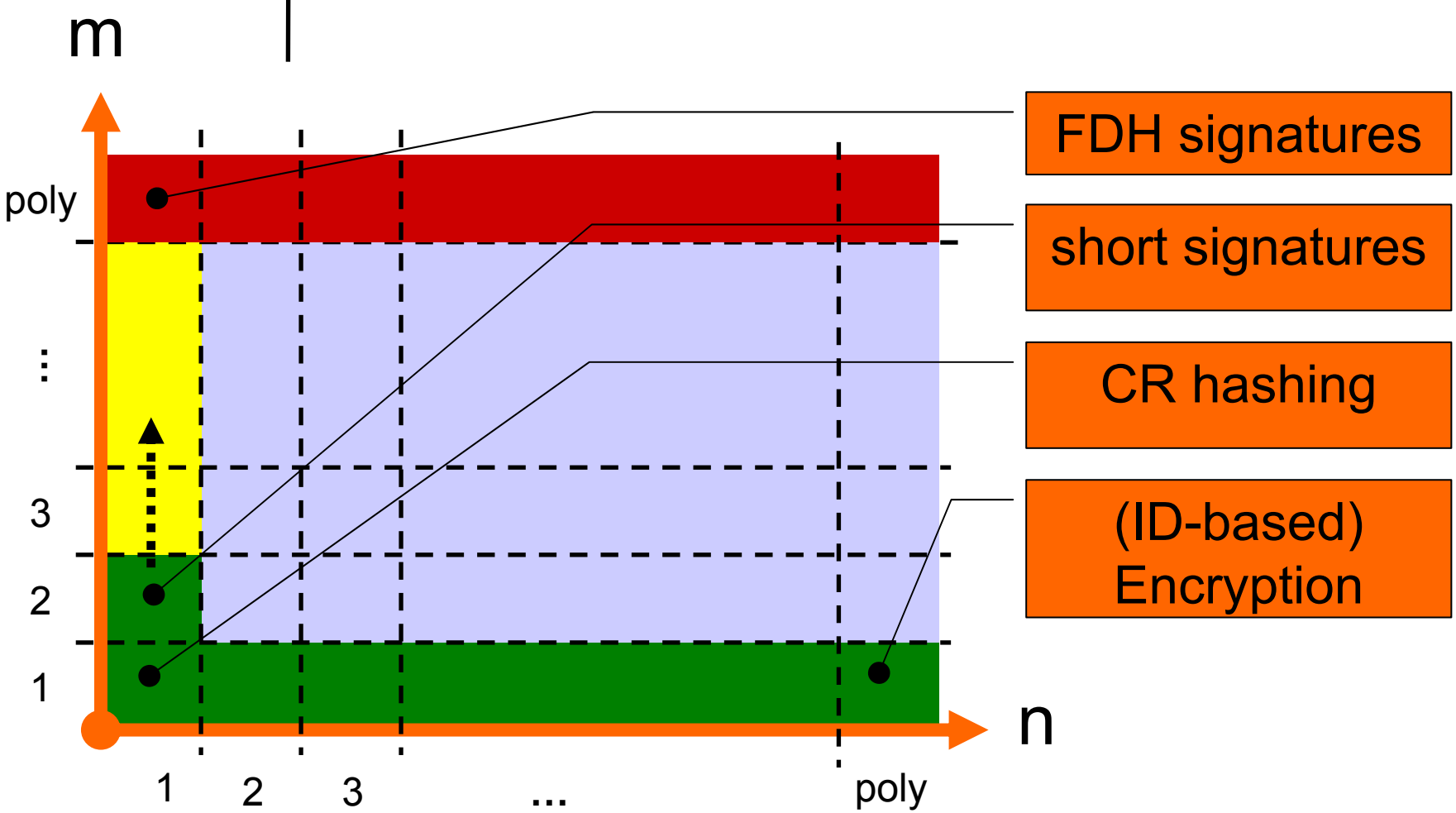
- Program $H_\kappa(X) = g^{a(X)}u^{b(X)}$ { **b(x)=0** : prob. p; **b(x)=random** : prob. 1-p

- $\Pr[b(X^{(i)}) = 0, i=1\ldots m \text{ and } b(Y) \neq 0] = p^m(1-p)$

Overview

1. Hash functions
2. Programmable hash functions
3. Applications of PHF's
4. Instantiations of PHF's

Applications of PHFs



Warm-up: collision resistant hashing

- **Theorem:**
  - If H=(1,1)-PHF and discrete log assumption holds in G, then H is collision resistant.
- **Proof:**
  - Assume adv finds any $X \neq Y$ with $H_\kappa(X)=H_\kappa(Y)$
  - Given (g,u), use TrapGen to setup key κ of (1,1)-PHF such that with non-negl. prob:
    - $H_\kappa(X) = g^{a(X)}$ and $H_\kappa(Y) = g^{a(Y)}\, u^{b(Y)}$, $b(Y)\neq 0$
    - $\Rightarrow g^{(a(X)+a(Y))/b(Y)} = u$ reveals $\log_g(u)$

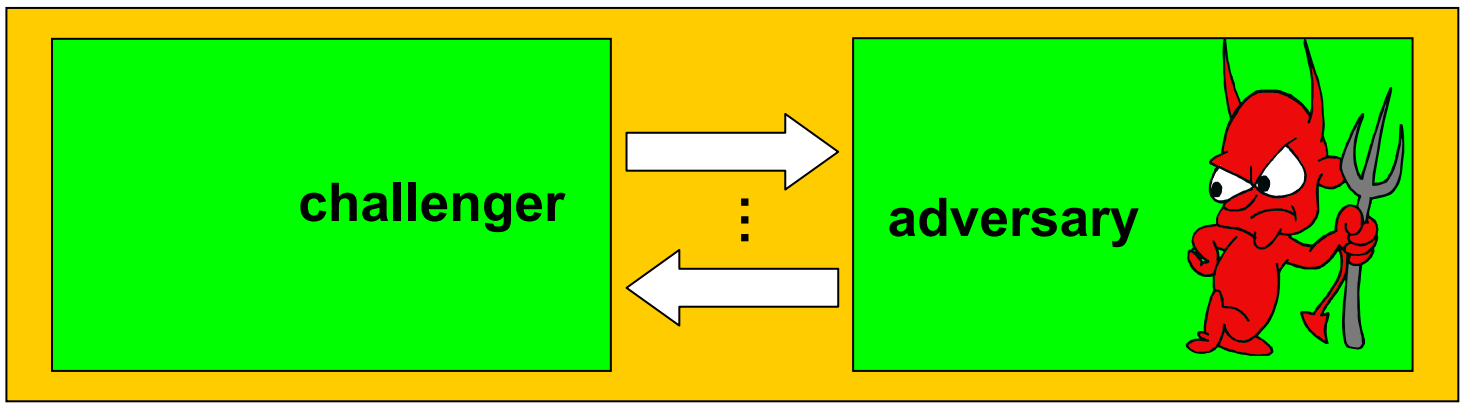
$$H_\kappa(X) = g^{a(X)}u^{b(X)}$$

Signatures

- **SIG=(Kg, Sign, Verify)**
- **Security: unforgeability against chosen-message attacks (UF-CMA)**

1. Adversary A gets pk
2. Ask for signatures σ on messages m (poly many)
3. Returns signature σ^* on message m^*

A wins if σ^* is valid sig on m^* and m^* not queried



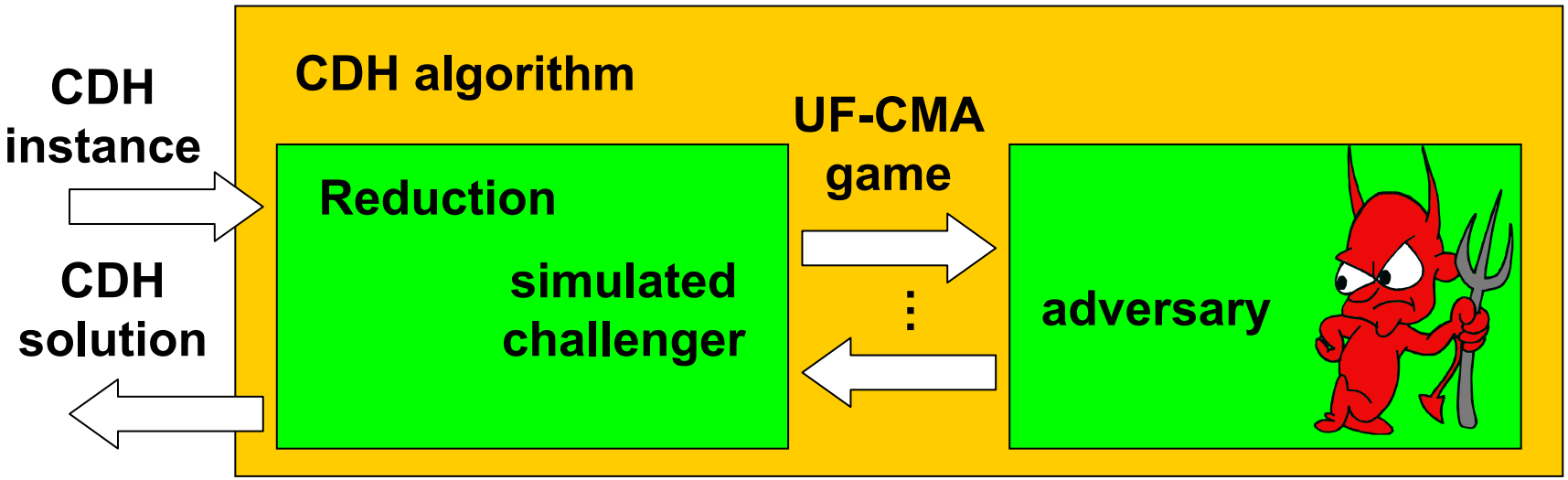
FDH signatures over pairings [BLS01]

- **G = prime-order group**
  - CDH assumption: Given g, g^x, g^y, computing g^{xy} is hard
  - pairing $e: G \times G \rightarrow G_T$
- **BLS signatures [BLS01]**
  - Public-key: g, $h=g^x$, $H_\kappa: \{0,1\}^* \rightarrow G$ hash function, κ from Kg
  - Secret-key: x
  - Sign(sk,m): $\sigma = H(m)^x \in G$
  - Verify using $h=g^x$ and pairing: $e(\sigma,g) = e(H(m),h)$?
- **Theorem:**

  H = (poly,1)-PHF and CDH assumption holds in G, then BLS signatures are UF-CMA secure
- **Signature size: 160 bits! ☺ (optimal!)**

Proof of theorem

- **Construct reduction algorithm that simulates the challenger in a UF-CMA attack and uses the adversary's forgery to solve the CDH instance**



Proof of theorem

- **CDH algorithm inputs (g, g^x, g^y), simulates:**
  - Setup
    - pk=g,h=g^x
    - H_κ with κ from TrapKg(g,u=g^y)
  - signature queries: $m_1, \ldots, m_q$, forgery: m*
    - Hope: $b(m_1)=\ldots=b(m_q)=0$ and $b(m^*)\neq 0$
    - H = (poly,1)-PHF $\Rightarrow$ Pr[hope true]=negl.
  - signature query(m): with b(m)=0:
    - $\sigma = H_\kappa(m)^x = (g^{a(m)})^x = h^{a(m)}$
  - forgery (m^*, σ^*) with $b(m^*)\neq 0$:
    - $\sigma^* = H_\kappa(m^*)^x = (g^{a(m^*)}\, u^{b(m^*)})^x = h^{a(m)}\,(g^{xy})^{b(m^*)}$
      $\Rightarrow$ CDH solution $g^{xy} = \sigma^*\, h^{-a(m)}$

$$H_\kappa(m) = g^{a(m)} u^{b(m)}$$

But….

H = random oracle ⇒ H = (poly,1)-PHF

Need scheme with (2,1)-PHF!

Short bilinear signatures [BLS01]

Theorem:
H = (m,1)-PHF and strong CDH assumption holds in G, then the signature scheme is secure

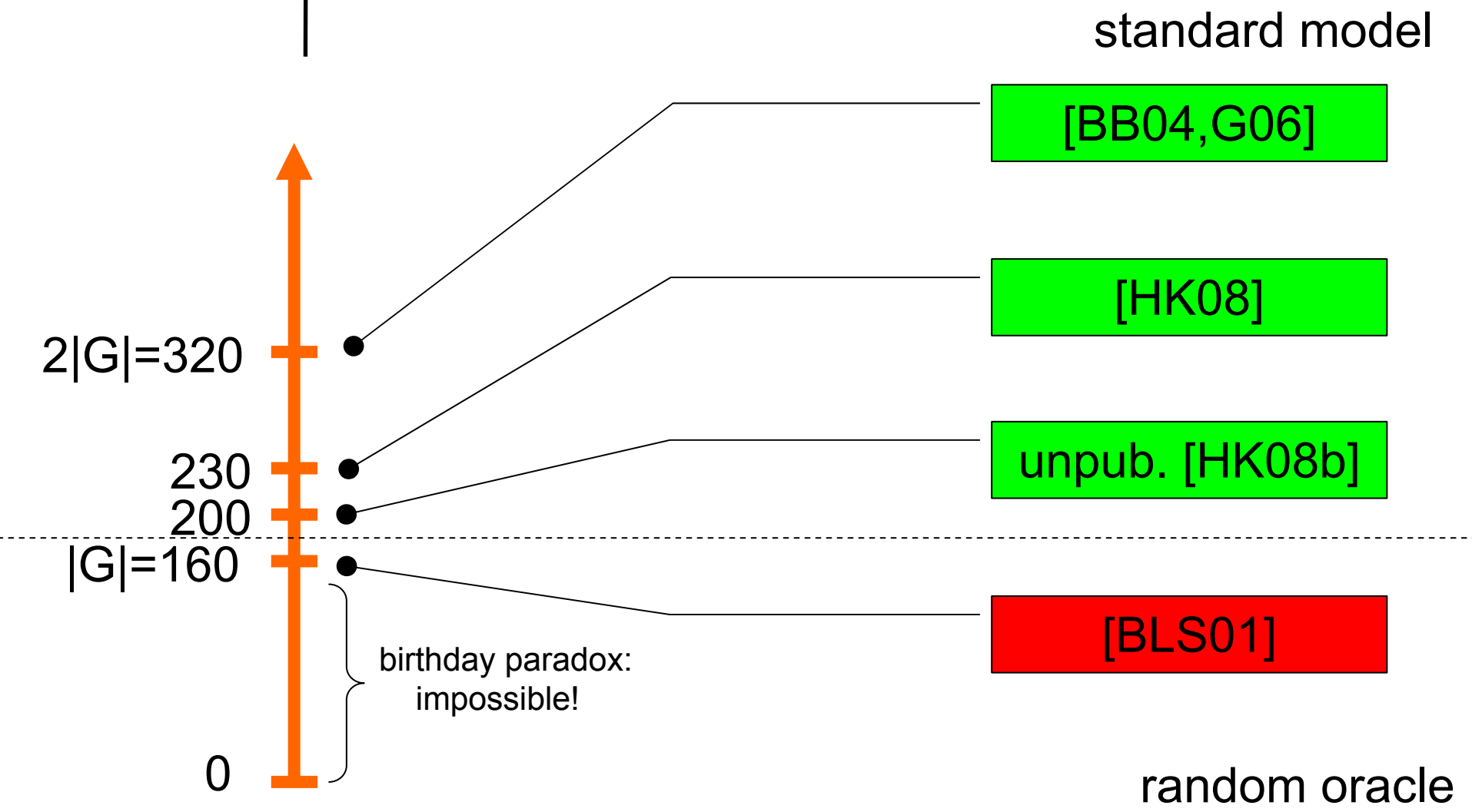
- Strong CDH assptn:
  - Given $g, g^x, g^{x^2}, g^{x^3}, \ldots$ computing $(c, g^{1/(x+c)})$ hard

Signature size:
- $|r| = 30 + 80/m$ bits sufficient
- $\Rightarrow$ Signature size 160+30+80/2=230 bits for m=2

- **Short signatures**
  - Public-key: g, $h=g^x$, $H_\kappa: \{0,1\}^* \rightarrow G$ hash function, κ from Kg
  - Secret-key: x
  - Sign(sk,m):
  - Verify using $h=g^x$ and pairing: $e(\sigma, h\, g^r) = e(H(m), g)$?

$$\sigma = (H(m)^{1/(x+r)}, r) \in (G \times \{0,1\}^{|r|})$$

Pairing based short signatures



Security proof I

uniform from $\{0,1\}^{|r|}$

- **Signature queries (m_i, σ_i),**

$$\sigma_i = (s_i = H_\kappa(m_i)^{1/(x+r_i)}, r_i)$$

- **Forgery (m^*, σ^*),**

$$\sigma^* = (s^* = H_\kappa(m_i)^{1/(x+r^*)}, r^*)$$

chosen by forger

- **Two cases**
  - **Easy** case: r^* new
  - **Hard** case: $r^* \in \{r_1, \ldots, r_q\}$

"adversary recycles randomness"

Security proof II

- **Hard case**: $r^* = r_{j1} = r_{j2} = \ldots = r_{jk}$
- **Simulation idea:**
  - given $g', g'^{x}, g'^{x^2}, g'^{x^3}$ setup of $g = g'^{\prod(x-r)}, u = g'^{\prod^*(x-r)}$:
    - setup: $(x+r_i)$th roots of $H_\kappa(m)$ for $r_i \neq r^*$ always easy
    - **red m_{ji}:** $(x+r^*)$th roots of $H_\kappa(m_{j1})$, …, $H(m_{jk})$ easy
    - **green m^*:** $(x+r^*)$th root of $H_\kappa(m^*)$ solves strong CDH

    } $(k,1)$-PHF
  - Corresponds to $(k,1)$-PHF property of H:
    "k times something possible, 1 time not"
  - Use $H_\kappa(m^*)^{1/(x+r^*)}$ to break strong CDH instance

$\prod$ over $r \neq r^*$

- **During simulation**
  - whp: at most k collisions in set $\{r_1, \ldots, r_q\}$

Security proof III

- **How many collisions in set $\{r_1, \ldots, r_q\}$?**
  - r_i uniform from $\{0,1\}^{|r|}$
  - R_k=Pr[k collisions in q uniform values $\in \{0,1\}^{|r|}$]
  - Extended "birthday paradox":
    $$R_k \leq q^k/2^{|r|(k-1)}$$
  - $\Rightarrow$ the bigger the k, the smaller the |r|:
    with (k,1)-PHF we can set |r|=30 + 80/m

Applications

Standard model

- Short Bilinear signatures
- Short RSA signatures
- Identity-based encryption
  - Waters' IBE
- Public-Key Encryption

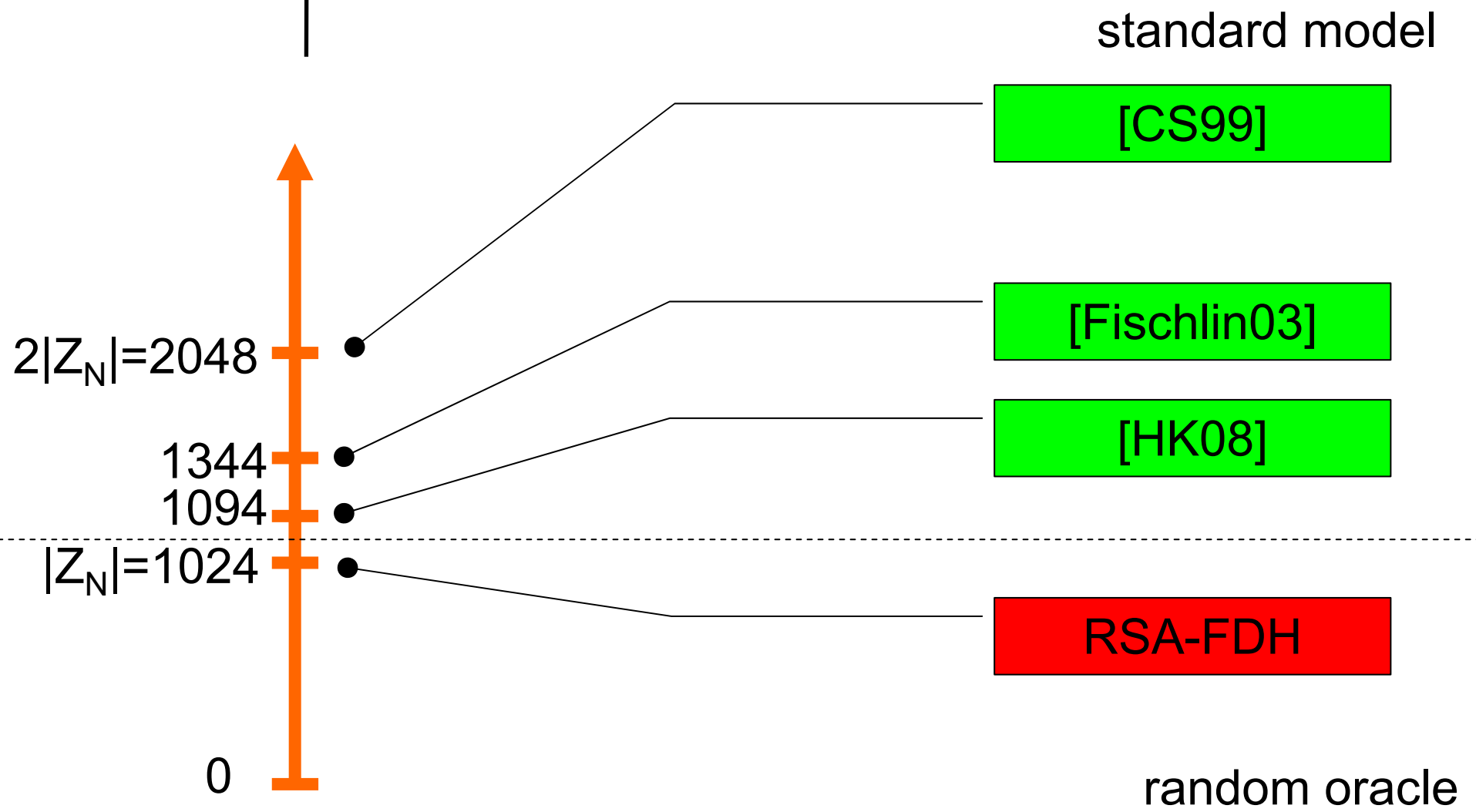
Random oracle model

- Full domain hash
- …

Short RSA signatures

- **N = pq RSA modulus**
  - Strong RSA assumption: Given N, z, computing $z^{1/e}$, e is hard
- **Short RSA signatures**
  - Public-key: N, H_κ: $\{0,1\}^* \rightarrow Z_N$ hash function, κ from Kg
  - Secret-key: p,q
  - Sign(sk,m): $\sigma = (H(m)^{1/e} \bmod N, e=\text{random prime} \in \{0,1\}^{|e|})$
  - Verify(pk,σ = (s,e)): $s^e = H(m)$?
- **Theorem:**
  - H = (m,1)-PHF and strong RSA assumption hold, then the signature scheme is secure
- **Signature size:**
  - Signature size 1024+30+80/2=1094 bits for m=2

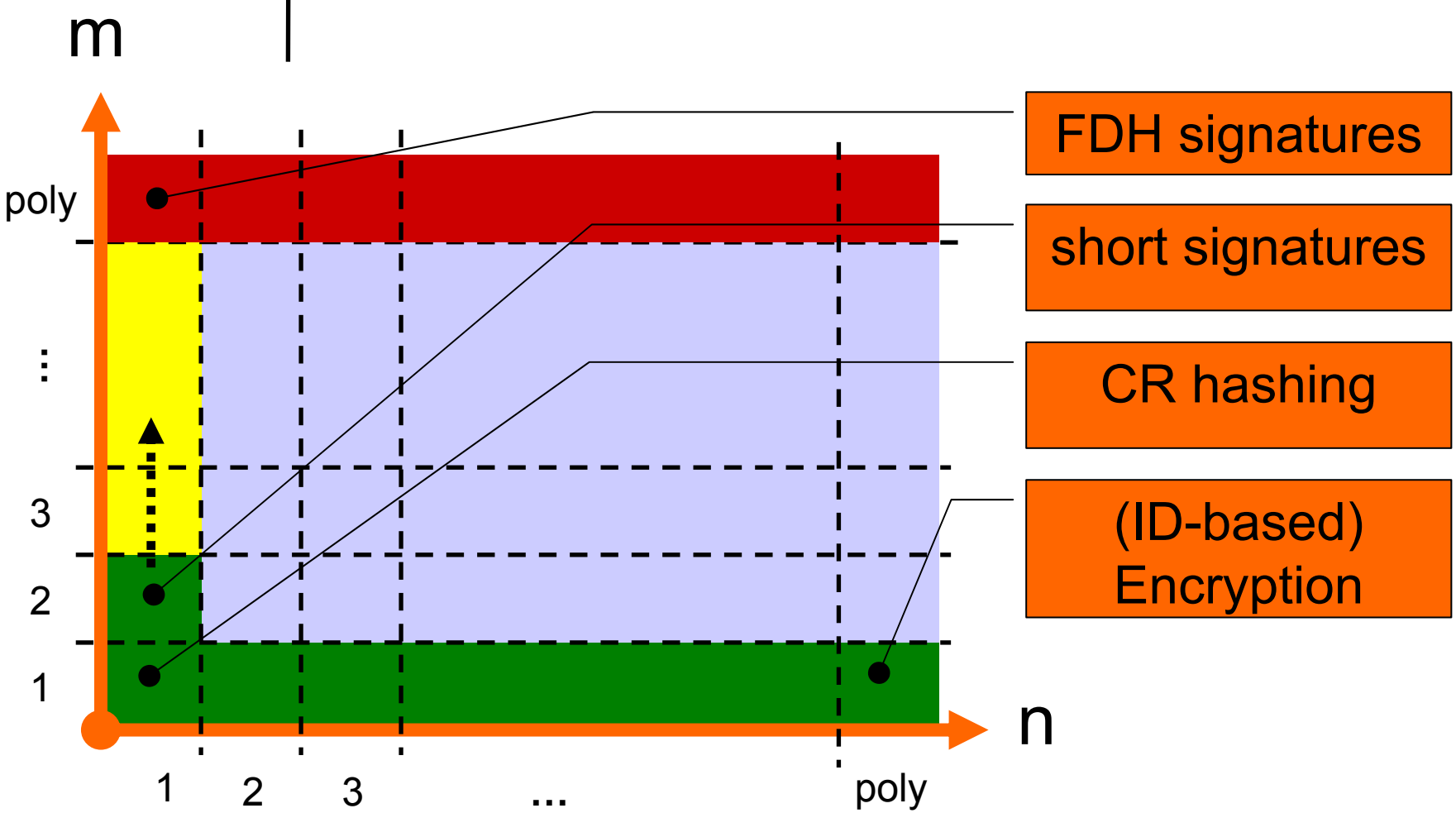
RSA based short signatures



Overview

1. Hash functions
2. Programmable hash functions
3. Applications of PHF's
4. Instantiations of PHF's

Instantiations



Multi-Generator hash function H_{MG}

- **Hash function from "goude eeuw"**
  [Chaum, Evertse, van de Graaf, 1987]
  - Kg:
    - hash key $\kappa = (h_0, \ldots, h_n) \in G^{n+1}$
  - Eval(κ, m):
    - H: $\{0,1\}^n \rightarrow G$,

$$H_{MG}(m) := h_0 \prod h_i^{mi}$$

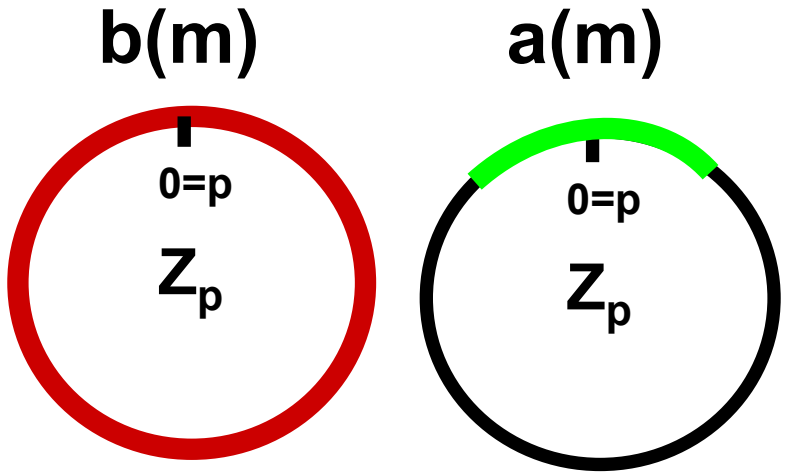
$m=(m_1,\ldots,m_n)$

- **Theorem:**
  - H_{MG} is (2,1)-PHF [with prob 1/n]
  - H_{MG} is (1,poly)-PHF [with prob $1/qn^{½}$]

Proof of (2,1)-PHF

- **Kg:**
  - key $\kappa = (h_0, \ldots, h_n) \in G^{n+1}$
- **Eval(κ, m):**
  - H: $\{0,1\}^n \rightarrow G$, $H_{MG}(m) := h_0 \prod h_i^{mi}$
- **TrapKg(g,u):**
  - trapdoor $t = (a_0, \ldots, a_n, b_0, \ldots, b_n)$
    - $b_i \in Z_p$ (random masks)
    - $a_i = \{-1,0,1\}$ random walk
  - key $\kappa = (h_0, \ldots, h_n)$,
    - $h_i := u^{ai} \cdot g^{bi}$
- **TrapEval(t,m):**
  - $a(m) = a_0 + \sum a_i m_i \in \{-(n+1), \ldots, (n+1)\}$ over Z!!
  - $b(m) = b_0 + \sum b_i m_i \in Z_p$

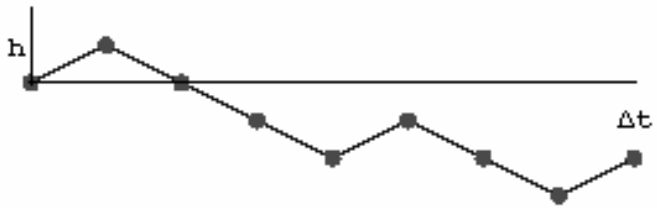
$$H_{MG}(m) = h_0 \prod h_i^{mi} = g^{b(m)} u^{a(m)}$$

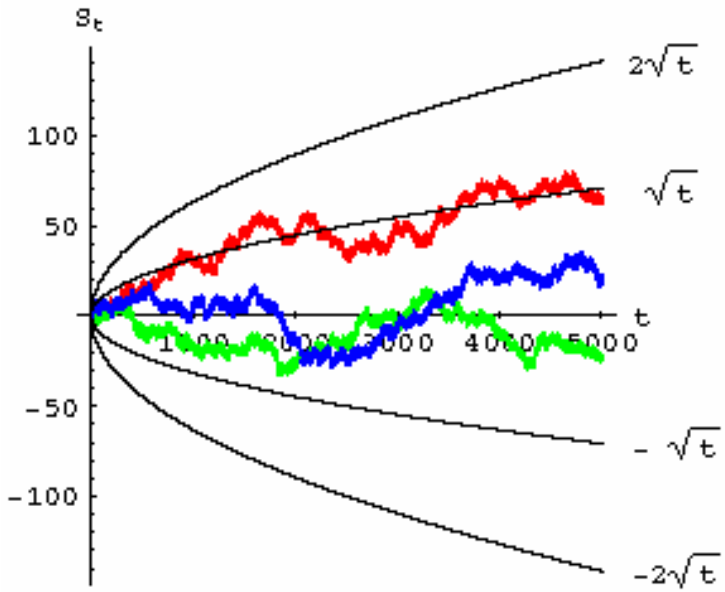


Distribution of a(m)?

Interlude: random walks

- **Bernoulli random walk of length n:**
  - $X_i \leftarrow_R \{-1,1\}$
  - $X = X_1 + \ldots X_n$



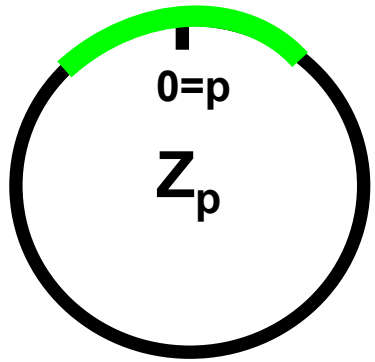


- **Random walk lemma:**
  for all $|t| \leq \sqrt{n}$: $\Pr[X = t] \approx 1/\sqrt{n}$

- **We need: special random walk with $X_i \leftarrow_R \{-1,0,1\}$**
  - essentially same behavior

Back to the proof

- **$a(X) = a_0 + \sum a_i X_i \in \{-(n+1), \ldots, (n+1)\}$ over Z**
  - a_i's are {-1,0,1} random walks
- **To show for (2,1)-PHF:**
  - For all strings X,X'≠Y:
    Pr[a(X)=a(X')=0 and a(Y)≠0] ≥ 1/n
- **Idea:**
  - for all X: a(X) random walk of length hw(X)+1
  - RW lemma: 1/sqrt(n) ≤ Pr[a(X) = 0] ≤ const
  - Pair-wise independence:
    for X,X': 1/sqrt(n) ≤ Pr[a(X) = 0 | a(X') = 0] ≤ const
  - Third requirement a(Y)≠0 ugly case distinction ☹



Open problems

- **Programmable hash functions**
  - (m,1)-PHF for m≥3?
  - (1,1) or (1,poly)-PHF with smaller description?
- **Extensions**
  - Formalize other useful hash properties?
    - Related: POWHFs, VRFs, non-malleability, seed incompressability…
  - What needed to prove OAEP, Schnorr, DSA, …?
- **Hash attacks**
  - Attack PHF property (or others) of MD-815