Workshop on Cryptography Fields Institute - Toronto / Nov. 27-Dec. 1, 2006

Efficient Pseudorandom Generators Based on the DDH Assumption (mixed with some other results)

Berry Schoenmakers

Joint work with **Andrey Sidorenko** and (partly) with **Reza Rezaeian Farashahi**

Coding & Crypto group

TU Eindhoven, The Netherlands

Plan

- Revisiting a basic problem:
  - Given a random bit source. Generate uniformly random numbers in a given interval. **New, simple algorithm**:
    - competitive in context of secure multiparty computation
    - cryptographic relevance witnessed by Bleichenbacher's attack on DSA
- Concrete security of provably secure PRGs
  - Focus on DL-related assumptions
  - **New construction based on k-DDHI** (k bounded Dec DH Inversion)
  - Intermezzo: **cryptanalysis of Dual Elliptic Curve Generator**
    - also done by Brown, and by Gjøsteen
- **New, simple PRGs based on the DDH problem**
  - First "practically" tight reduction to DDH (except for loss due to hybrid lemma)
  - DDH is as strong as DL (and DH) assumption, in practice => good concrete security, hence good performance
  - Specific instances
    - QR(p): group of quadratic residues
    - G_q: arbitrary subgroup of Z_p^*

Random numbers in [0,B), $2^{n-1}<B<2^n$

- Given a source of (uniform) random bits.
- Two folklore algorithms for generating $x \in [0,B)$.

Alg.1: pick $x \in \{0,1\}^n$, using n random bits, until $x<B$

Alg.2: pick $x \in \{0,1\}^{n+k}$, using n+k random bits; output x mod B

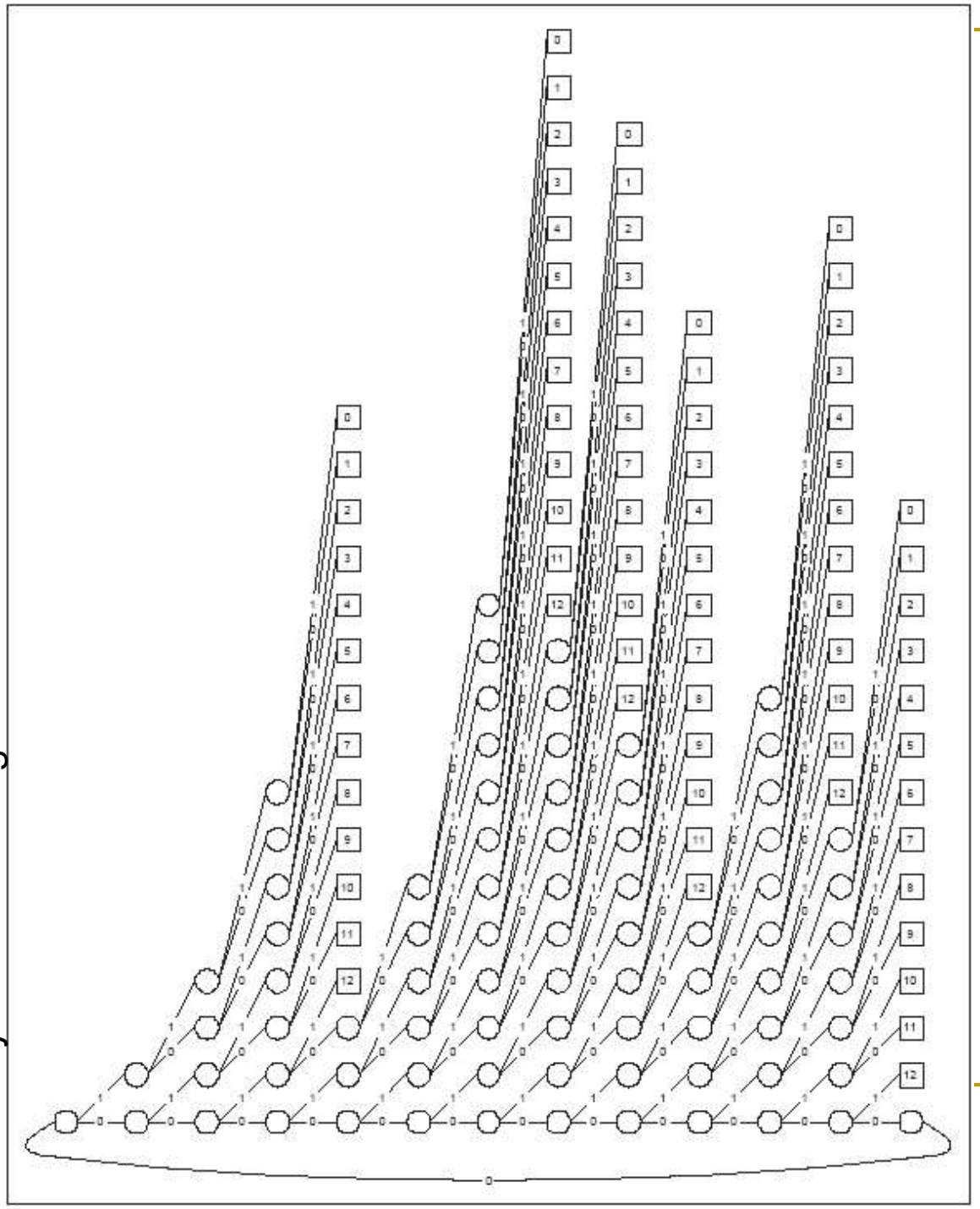
- Properties:
  - Alg.1: perfectly uniform; but wastes up to n bits on average (worst case $B = 2^{n-1} + 1$); Las Vegas algorithm
  - Alg.2: statistical distance $\Delta<1/2^k$; wastes k bits exactly

Our algorithm

- Generate random x ∈ [0,B) bit by bit, starting from the most significant bit, comparing with most significant bits of B-1.
- Algorithm: let x_i be next random bit
  - if $x_i > (B-1)_i$, start all over "too large"
  - if $x_i = (B-1)_i$, continue with next bit "unsure"
  - if $x_i < (B-1)_i$, complete x with random bits and stop "home free"
- Randomness complexity: n bits plus some waste.
- Question: what's the waste?

Analysis of randomness complexity

- 1st computing the exact probability distribution and then the expected value is cumbersome
- We determine expected value E directly!
- Example: $S = \Sigma_{i=0..\infty} r^i$:
  - $S = \Sigma_{i=0..\infty} r^i = 1 + \Sigma_{i=0..\infty} r^{i+1} = 1 + r\,S$, so $S = 1/(1-r)$
- Example: $T = \Sigma_{i=0..\infty} i\, r^i$:
  - $T = \Sigma_{i=0..\infty} (1+i)\, r^{i+1} = r\,S + r\,T$, so $T = r/(1-r)^2$
- By conditioning on the right event, this leads to:
  - **$E = n + 2^n/B \;\; \Sigma_{i=2..n}\, i(1-(B-1)_{n-i})/2^i \;< n + 3$**
  - So, waste is bounded by a small constant!
  - Averaged over all B, waste is approx. 1.11 random bits



Courtesy Benne de Weger

Knuth-Yao 1976:
minimize randomness
complexity

Only wastes approx.
0.58 random bits
on average (over all B)
and <1 random bit
in the worst case.

Example:
Given random bits.

Generate random
integers mod 13

Context of secure multiparty computation

- Knuth-Yao actually prove that any probability distribution can be generated from random bits, wasting < 2 bits (on average).

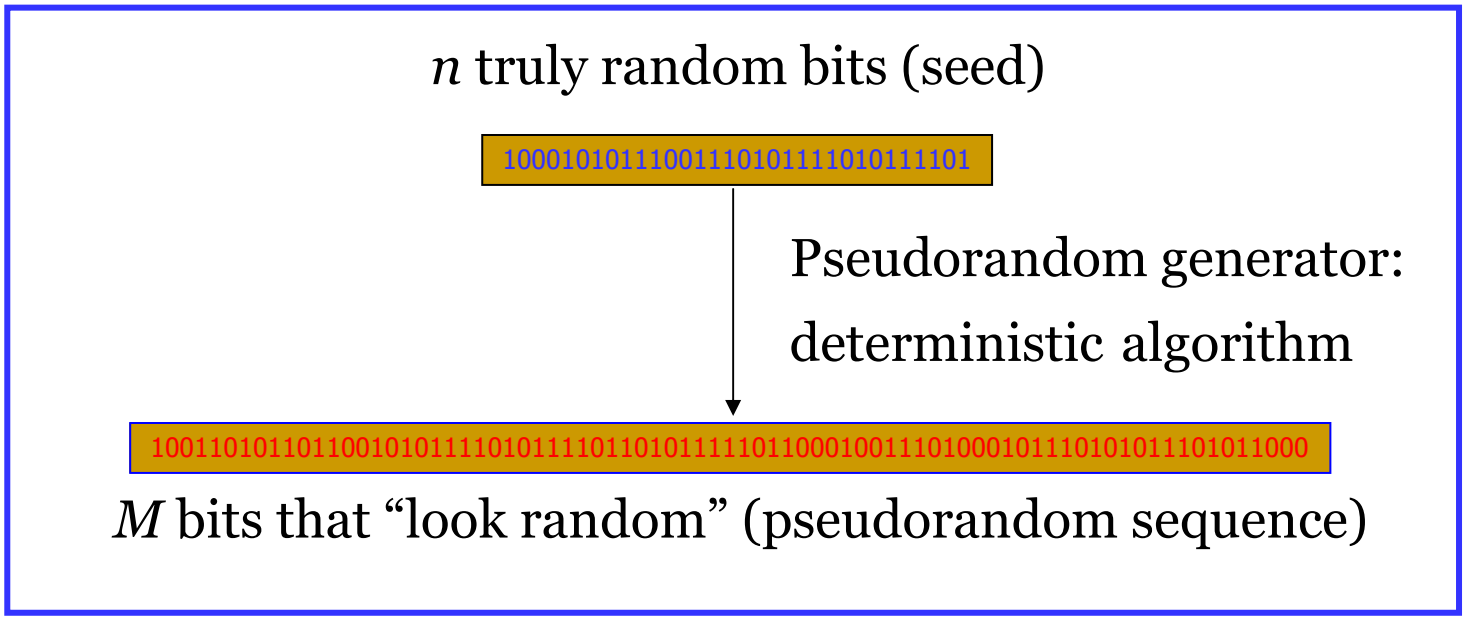
- In context of secure multiparty computation:
  - Generating random bits is expensive.
  - But, also comparing bits, arithmetic with bits, etc.

- Our algorithm (and variants) strike a better balance than Knuth-Yao's minimal waste algorithm, depending on the setting
  - cheaper to make it oblivious

And, the other way around …

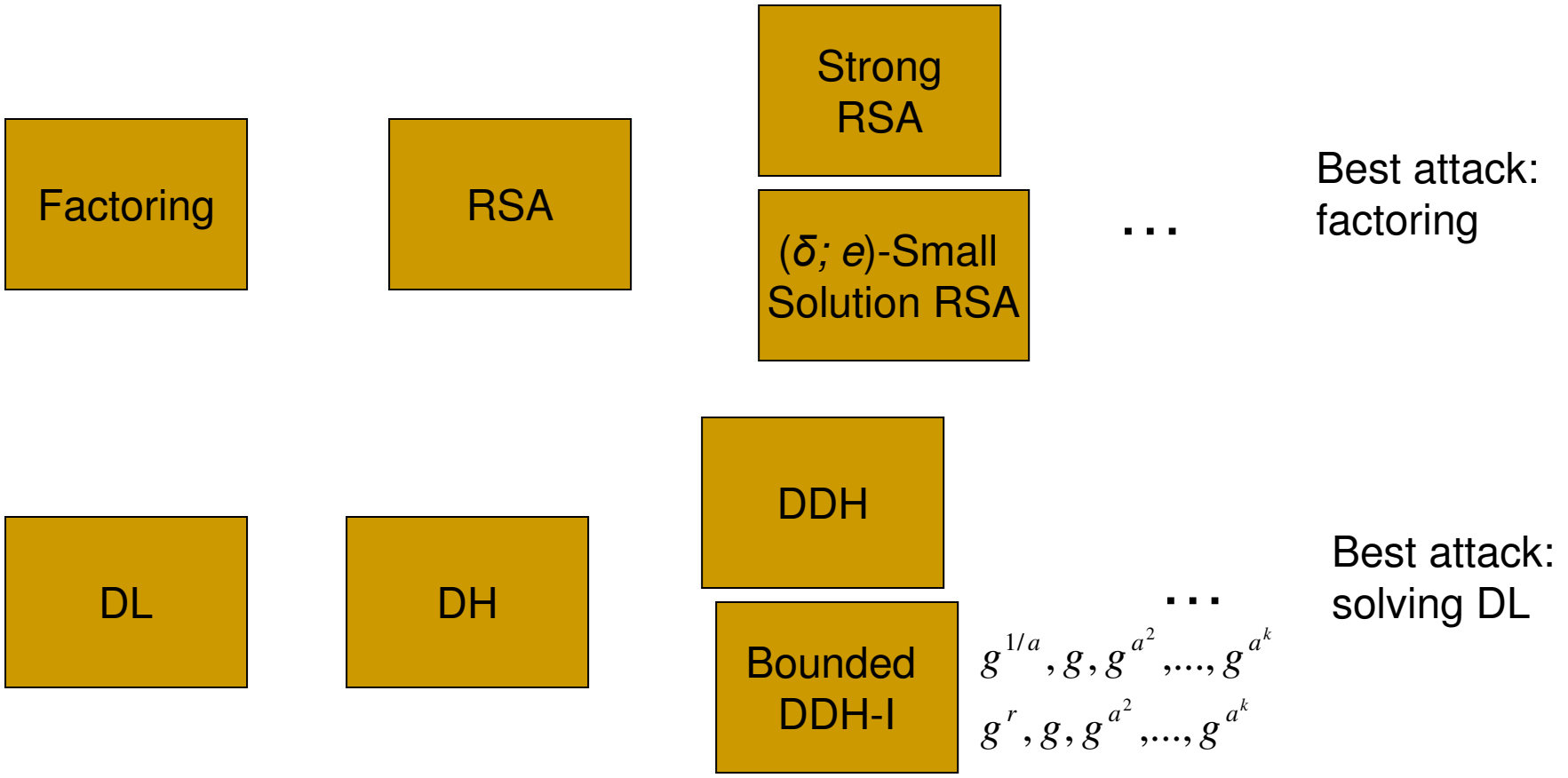
- Given a random numbers x ∈ [0,B).
- How to extract as many random bits from x?
- Alg. (we found this in Barker-Kelsey 2005):
  - compare bits of x and B, starting at most significant bit, until difference is found.
  - output all remaining bits of x, after position where difference occurs.
- Barker-Kelsey give no analysis
- We find: **$E \geq n - 2 + n/(2^n-1) > n - 2$**, for every bound B

Pseudorandom Generator (PRG)



- Preferably M>>n and a fast PRG
- ***Focus on provably secure PRGs***
  - a PRG is called provably secure if "breaking" the PRG is as hard as solving a notoriously hard problem

Strong Assumptions at a Bargain !



Theoretically: **different** assumptions (for all we know …)
Practically: **equivalent to** factoring and DL, respectively

Provably secure PRGs

Pseudorandom sequence for a truly random seed

Distinguisher: running time at most T

"I think that you gave me a pseudorandom sequence"

OR

Truly random sequence of the same length

If probability of successful guess $< \frac{1}{2} + \varepsilon/2$ the PRG is (T, ε)-secure

Provably secure PRGs (cont.)

(T, ε)-distinguisher for a PRG: $\{0,1\}^n \rightarrow \{0,1\}^M$

reduction →

(T', ε')-solver for a hard problem with security parameter n
(e.g., DL problem in n-bit finite field)

- If $T'/\varepsilon' \approx T/\varepsilon$, reduction is tight
- If $T'/\varepsilon' >> T/\varepsilon$, reduction is not tight

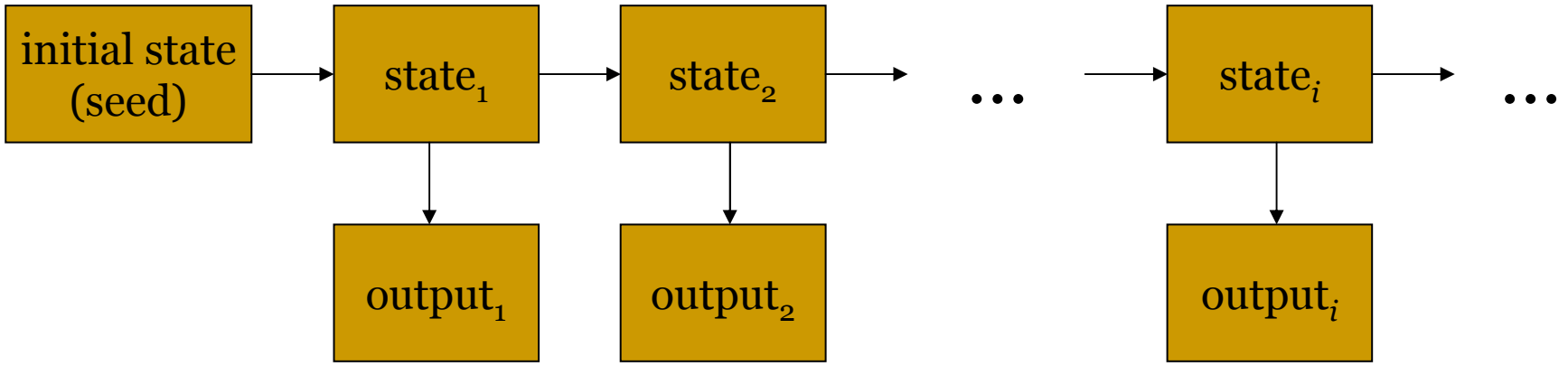
- If the reduction is tight, a desired security level can be achieved for a relatively low value of security parameter n

Security of PRG: formal definition

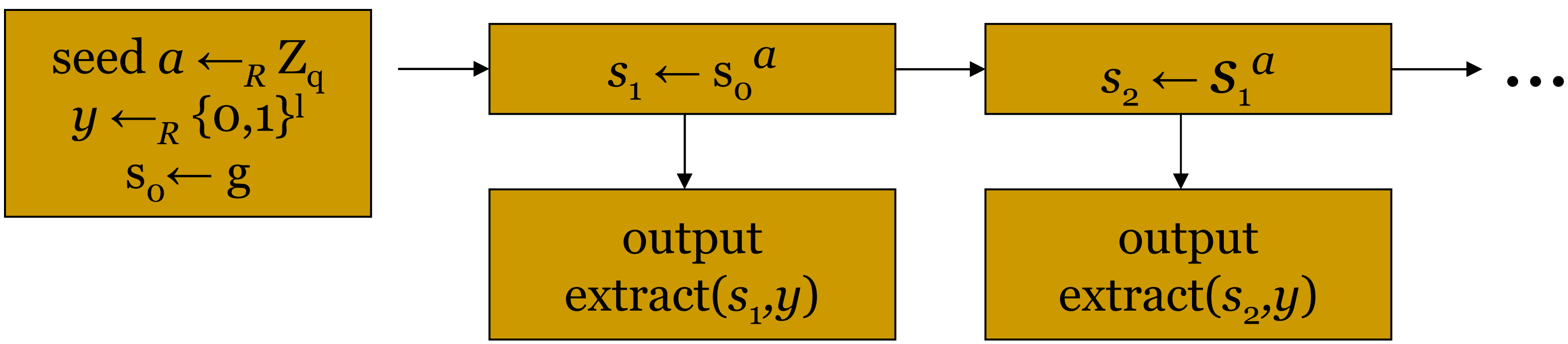
- Pseudorandom generator PRG: $\{0,1\}^n \rightarrow \{0, 1\}^M$
- Distinguisher D: $\{0,1\}^M \rightarrow \{0, 1\}$
- Denote by U_l uniform distribution on $\{0, 1\}^{2^l}$, $l > 0$
- PRG is called (T, ε)-secure if for all T-time distinguishers D

$$| \Pr[D(\mathrm{PRG}(U_n)) = 1] - \Pr[D(U_M) = 1] | < \varepsilon$$

Typical PRG


initial state (seed)
state$_1$
state$_2$
...
state$_i$
...
output$_1$
output$_2$
output$_i$

k-DDHI based PRG

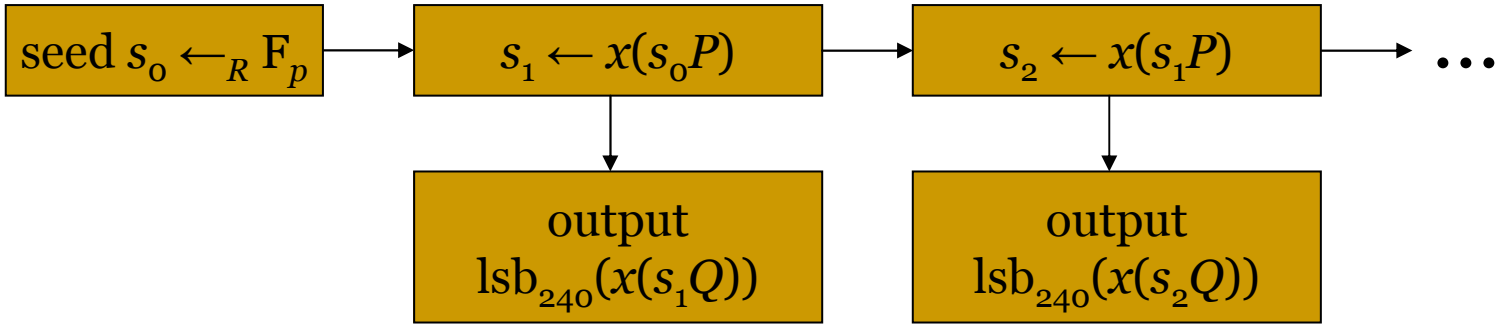


- Universal hash function used as extractor
- Good results, but assumption k-DDHI less standard
- k-DDHI: distinguish

$$\begin{matrix} g^{1/a}, g, g^{a^2}, \ldots, g^{a^k} \\ g^{r}, g, g^{a^2}, \ldots, g^{a^k} \end{matrix}$$

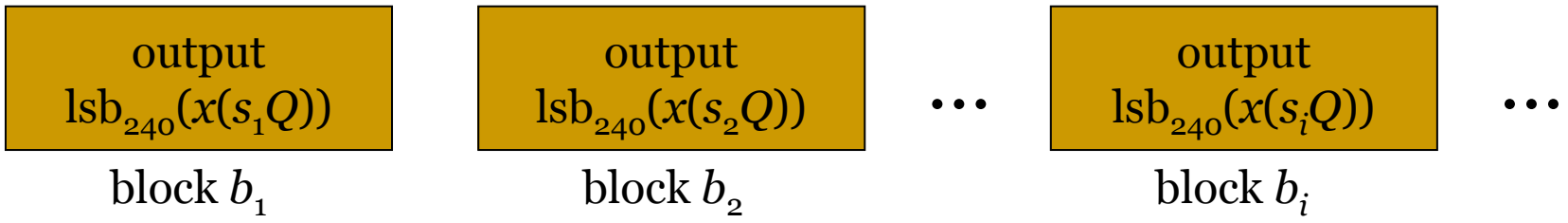
Dual Elliptic Curve PRG

- Proposed by Barker and Kelsey in a NIST draft standard [BK05]
- For prime $p = 2^{256} + 2^{224} + 2^{192} + 2^{96}+1$, let $E(F_p)$ be an elliptic curve such that $\#E(F_p)$ is prime. Let $P, Q \leftarrow_R E(F_p)$



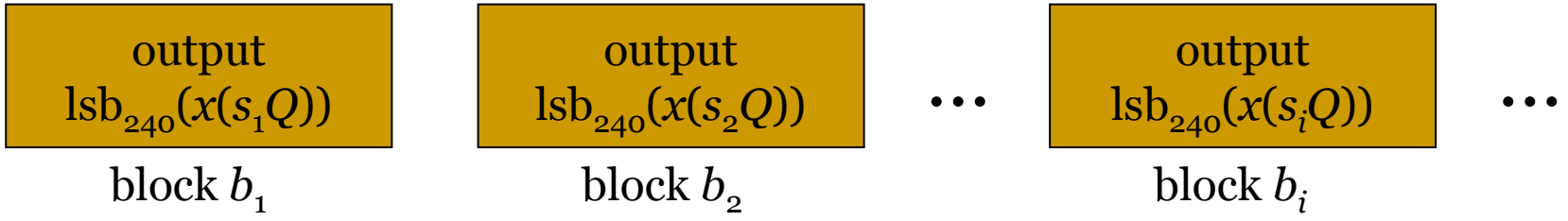
- sequence $s_i Q$ is indistinguishable from sequence of uniformly random points under DDH assumption and *x*-logarithm assumption [Brown06]
- however, random bits are extracted from random points improperly so the PRG is insecure [G06, SS06]

Distinguishing attack



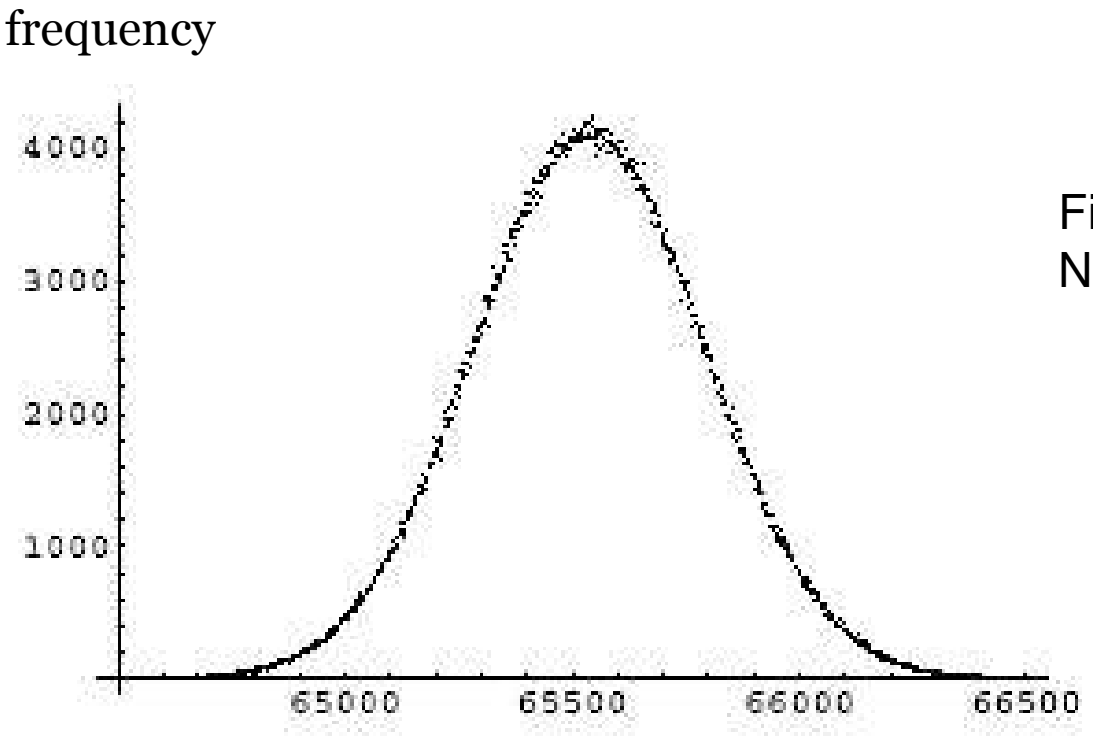
- Point s_iQ is mapped to output block $\text{lsb}_{240}(x(s_iQ))$
- Output blocks with more preimages show up more often
- Blocks $b \leftarrow_R \{0, 1\}^{240}$ have on average #E/(# of blocks) ≈ $2^{256}/2^{240} = 2^{16}$ preimages
- Blocks $\text{lsb}_{240}(x(R))$ with $R \leftarrow_R E(F_p)$ have on average more than 2^{16} preimages
- Thus, blocks $\text{lsb}_{240}(x(s_iQ))$ have on average more than 2^{16} preimages

The distinguishing attack is as follows…



- For each output block b_i count the number of preimages, i.e., count the number of points P such that $b_i = \mathrm{lsb}_{240}(x(P))$
- If the average number of preimages is above 2^{16}, decide that the sequence is produced by the PRG;
- Otherwise, decide that the sequence is "truly random"

Simulation



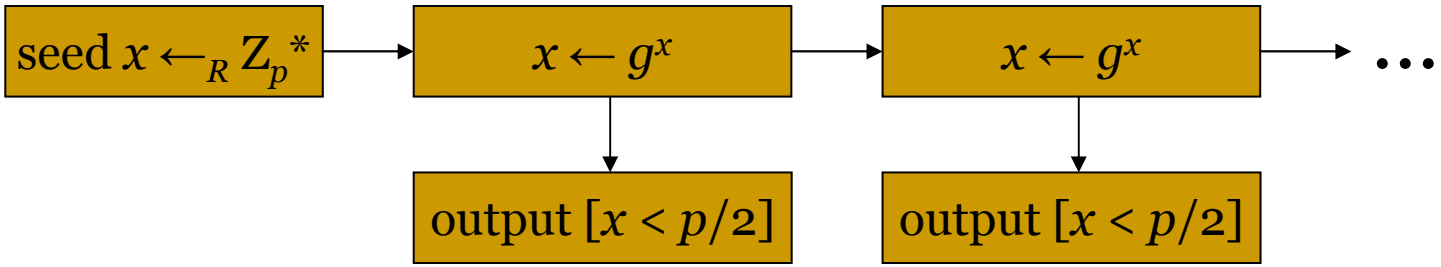
Fits normal distribution
N(65537.0, 255.6)

$2^{16} = 65536$

- 330 files produced by Dual Elliptic Curve PRG have been tested
  - each file consists of 4000 output blocks
- In 59% of files the average number of preimages is above 2^{16}
- Running time of the attack is about 3 hours on a 3GHz Linux machine with 1Gb of memory

Blum-Micali PRG

- Proposed by Blum and Micali [BM84], based on DL-problem in Z_p^*
- Provably secure



- Outputs only 1 bit per modular exponentiation
- Let $n = \log_2 p$. Suppose the BM PRG is not (T, ε)-secure. Then the DL problem can be solved in time $T' = 64\, n^3\, (M/\varepsilon)^4\, T$ with success probability $\varepsilon' = 1/2$
- $T'/\varepsilon' >> T/\varepsilon$, so reduction is not tight

Blum-Micali PRG (cont.)

- All in all, BM PRG is (T, ε)-secure if

polynomial in n

subexponential in n

$$128\, n^3\, (M/\varepsilon)^4\, T < T_{DL}(Z_p^*)$$

where

$$T_{DL}(Z_p^*) = a\, \mathrm{Exp}[1.9229\, (n \ln 2)^{1/3} (\ln (n \ln 2))^{2/3}],$$
$$a \approx 4.7 \cdot 10^{-5} \text{ time units (DES encryptions)}$$

- For $M = 2^{20}$, $T/\varepsilon = 2^{80}$, BM PRG is (T, ε)-secure if $n >$ 61000
- High seed length n implies poor efficiency
  - the cause is a far from tight reduction
- We propose a PRG with a much better security reduction
  - based on the DDH assumption (stronger than DL assumption)
  - output of n bits per iteration

Decisional Diffie-Hellman (DDH) problem

- $G=\langle g \rangle$ is a multiplicative group of prime order q
- Algorithm A solves the DDH problem in G with advantage ε iff for a random triple (a, b, r)

$$| \Pr(A(g, g^a, g^b, g^{ab}) = 1) - \Pr(A(g, g^a, g^b, g^r) = 1) | \geq \varepsilon$$

- For concrete analysis:
  - **DDH problem is assumed to be as hard as the DL problem**

DDH generator (intuition)

- $G=\langle g \rangle$ multiplicative group of prime order q
- *Let* $a \leftarrow Z_q$ be a fixed integer

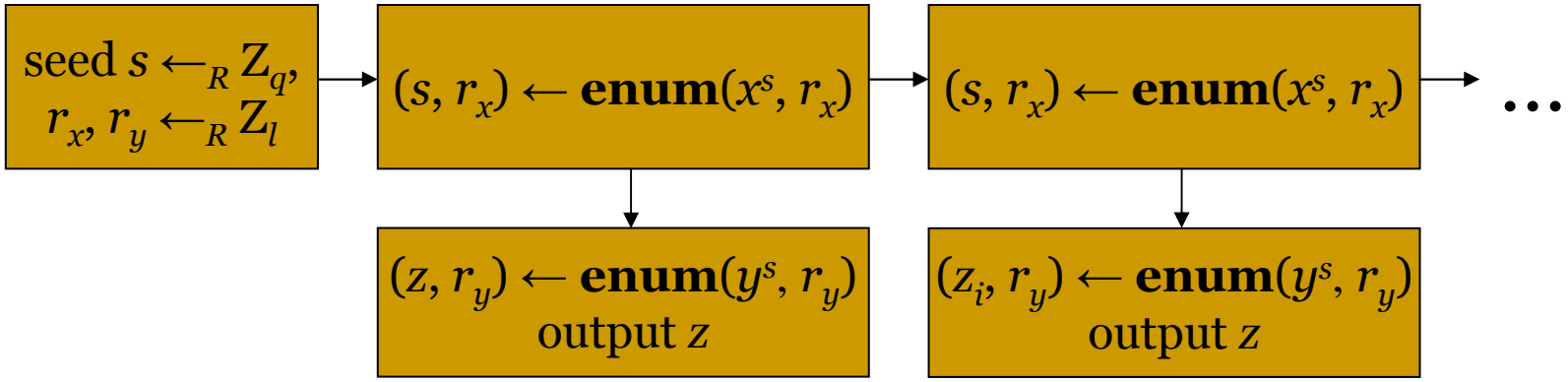
- Let **Double**$_{g,a}(b) = (g^b, g^{ab})$ [NR97]
  - for unknown $b \leftarrow_R Z_q$ the output is pseudorandom under the DDH assumption in G
  - "doubles" the input
- Is **Double** a pseudorandom generator?
  - No! It produces pseudorandom group elements rather than pseudorandom bits
  - Converting group elements into bits is a non-trivial problem
    - **Double** cannot be iterated to produce as much randomness as required by the application

DDH generator (construction)

- **enum** is a bijection "enumerating " the elements of group G:

$$\textbf{enum}: G \times Z_l \rightarrow Z_q \times Z_l$$

- Public parameters for DDH generator: $x, y \leftarrow_R G$



- Outputs $|q| = \log_2 q$ pseudorandom bits per step
- Seed length $|q| + 2|l|$

Security of the DDH generator

- DDH generator produces pseudorandom integers from Z_q
  - if q approx. 2^n then it produces pseudorandom bits directly
  - for an arbitrary q, additional effort has to be made to convert random numbers into random bits (from [0,q) to bits)

Theorem. Assume that $0 < (2^n - q)/2^n < \delta$ (for simplicity). Then (T, ε)-distinguisher for the DDH generator implies (T, $n\varepsilon/M - \delta$)-solver for the DDH problem in G

  - proof is based on the hybrid argument

Proof idea

- Given a 4-tuple (x,y,X,Y)
- Hybrid $H_j = (u_1, u_2, \ldots, u_{j-1}, \text{output}_1, \ldots, \text{output}_{k-j+1})$
  $= (v_1, v_2, \ldots, v_k)$ k=M/n

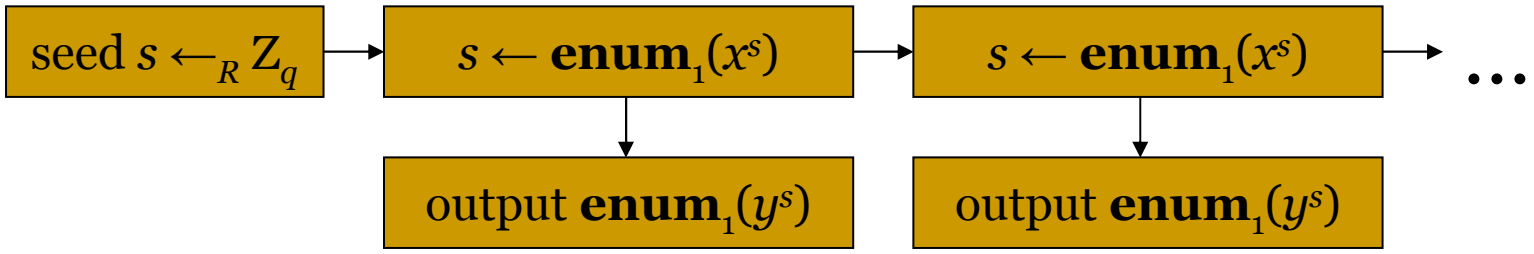
- Solver generates hybrids:
  - Pick j at random.
  - Pick random $v_1, v_2, \ldots, v_{j-1}$. Pick rx_0, ry_0 at random.
  - Set $(s_1, rx_1) = \text{enum}(X, rx_0)$
  - Set $(v_j, ry_1) = \text{enum}(Y, ry_0)$
  - Continue as in PRG to produce $(v_{j+1}, \ldots, v_k)$
- (x,y,X,Y) is DDH tuple iff output ~ H_j (else H_{j+1})

PRG1: instance based on QR(p)

- p is a safe prime: $p = 2q + 1$, q prime
- G = QR(p), $|G| = q$
- There exists a bijection from G to Z_q (Chevassut et al. 2005; Cramer-Shoup 2003, and ... ?):

$$\mathbf{enum}_1(x) = \begin{cases} x, \text{ if } x \le q; \\ p - x, \text{ if } x > q. \end{cases}$$

- Public parameters x, $y \leftarrow_R G$



- Extracts n bits per iteration (2 modular exponentiations)

PRG_1 (cont.)

- What seed length, *n*, guarantees security?
  - recall that for Blum-Micali PRG $n > 61000$
- Assume that $0 < (2^n - q)/2^n < n\varepsilon/2M$ (q close to 2^n)
- PRG_1 is (T, ε)-secure if $2MT/n\varepsilon < T_{DL}(QR_p)$
- For $M = 2^{20}$, $T/\varepsilon = 2^{80}$, PRG_1 is (T, ε)-secure if...
  $n > 1600$
- The seed length *n* is short because the reduction is (almost) tight
  - PRG_1 is much more efficient than Blum-Micali PRG
  - PRG_1 is based on a stronger assumption (the DDH assumption)
  - Limitation: works only for specific subgroup of Z_p^*

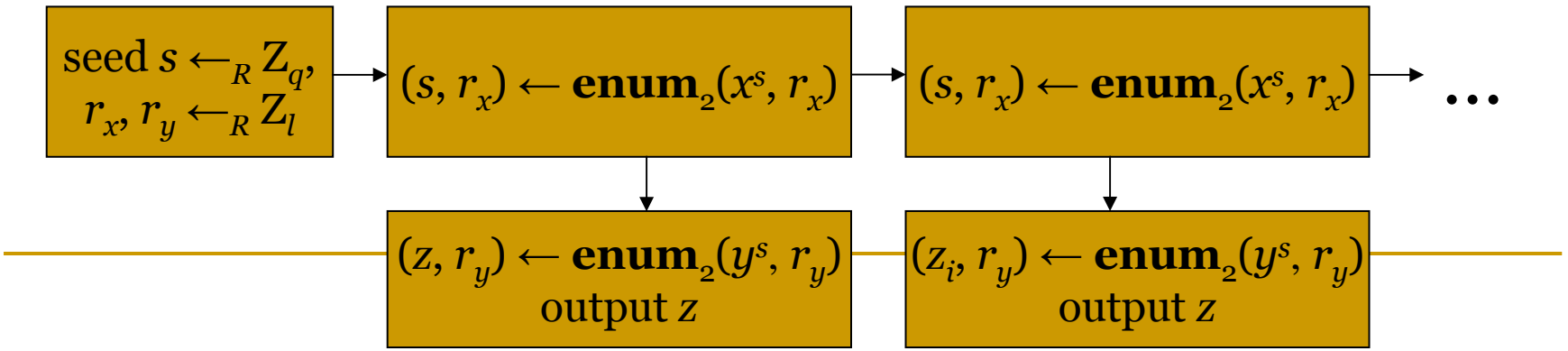
PRG_2: instance based on any subgroup

- p is a prime
- G is a (prime) order subgroup of Z_p^*, $|G| = q$, $(p - 1) = ql$
- t is an element of Z_p^* of order l, so $t^l = 1$

- Let $\mathbf{enum}_2$: $G \times Z_l \rightarrow Z_q \times Z_l$ be the following **bijection**:

$$\mathbf{enum}_2(x, r) = (x\, t^r \ \mathbf{mod}\ q,\ x\, t^r \ \mathbf{div}\ q)$$

- Public parameters $x, y \leftarrow_R G$



Conclusions

- General, simple construction of PRGs based on DDH assumption
- Two specific instances of the new PRG are presented
  - subgroup of quadratic residues modulo prime p – seed length |p|
  - arbitrary order q subgroup of Z_p^* -- seed length 2|p| -|q|
- Secure parameter n=|p| is about the same for PRG1 and PRG 2:
  - $n \approx 1600$
- Open problem: how to use an elliptic curve group?
  - would result in considerably shorter seeds
- For more details see http://eprint.iacr.org/2006/321