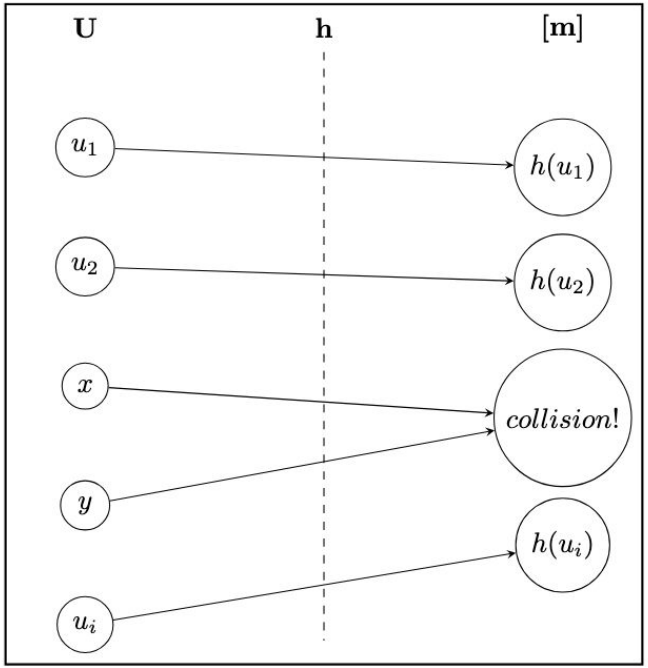
A Review of Damgård's Hash Function & Cryptanalysis

Caleb Alexander

Hash Function Background

A hash function is a function h : U → [m], which achieves a suitable level of randomness.



Hash Function Background

Important Properties Include:

- Universality
- Strong Universality
- Pre-Image Resistance
- 2nd Pre-Image Resistance
- Collision Resistance

Note that:

Strong Universality $\Rightarrow$ Universality

Collision Resistance $\Rightarrow$ 2nd Pre-Image Resistance $\Rightarrow$ Pre-Image Resistance

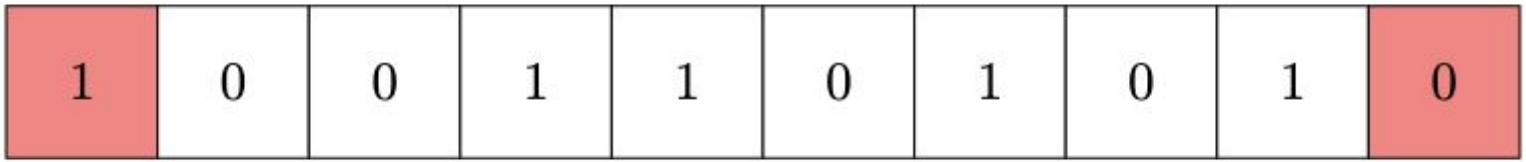
Hash Function Background

We will focus on collision resistance. Which we define as follows:

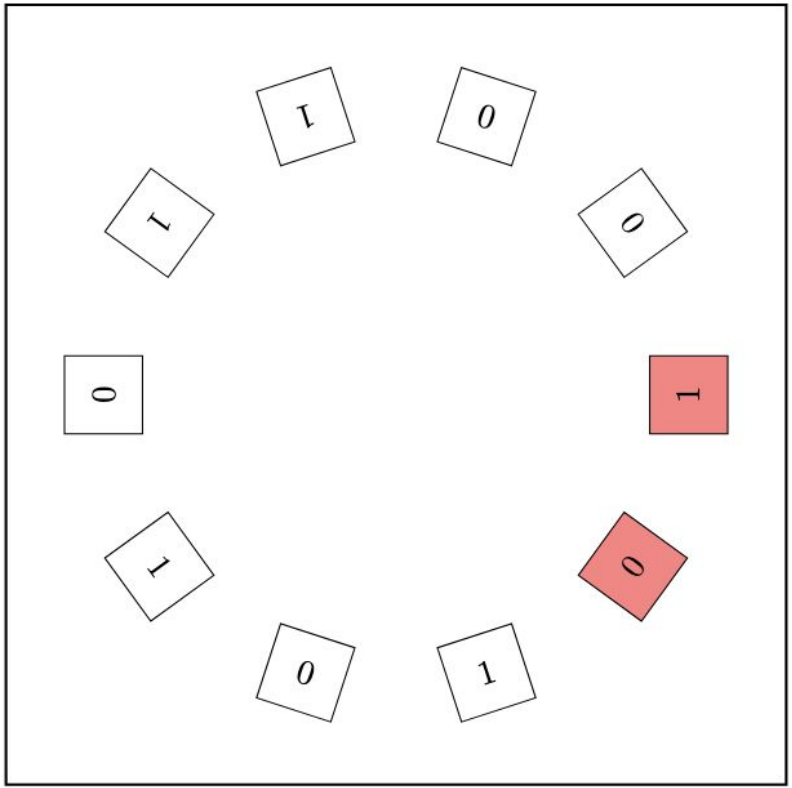
A hash function h, is said to be collision resistant iff it is computationally infeasible to find any x,y $\in$ U, so that h(x)=h(y).

Notice that computationally infeasible is a fuzzy definition.

Damgård's Function & Cellular Automata



Damgård's Function & Cellular Automata



Damgård's Function & Cellular Automata

The rule, introduced by Wolfram is

$$r(x_i) = x_{i-1} \oplus (x_i \vee x_{i+1})$$

1	0	0	1	1	0	1	0	1	0
1	1	1	1	0	0	1	0	1	0

Wolfram's Generator

1. Choose $x \in \{0, 1\}^n$, where n is the length of our board.
2. Compute $r(x_i)$ for all $i \in \{0, 1, \ldots, n - 1\}$. (i.e. evolve our board).
3. Take the first pseudo random bit to be our new x_0.
4. Repeat the process as many times as needed for as many bits as needed.

Damgård's Function & Cellular Automata

We will choose natural numbers c and d with c<d, and let $b_i(x)$ be the ith pseudo random bit returned by wolfram's generator. Then we define

$$f_0(x) = b_c(x), b_{c+1}(x), \dots , b_d(x).$$

But there are some insecurities, which depend on our choice of x, thus (assuming we want an input of length n) we will choose a bitstring, z, which is of length r < n where r is most secure being exactly n/2. Then our input will be the concatenation of x and z (x||z). So our final function is

$$f(x) = f_0(x||z).$$

Damgård's Function & Cellular Automata

Statistical testing, showed wolfram's PRG to be relatively safe. However, no proof was ever given for its safety against polynomial-time enemies.

Damgård showed that for his hash function, if $f^t(x) = f^t(y) = z$, and 2t consecutive bits of x and y were equal, then x = y.

Pf: Let j be the bit directly to the right of the 2t bits which we know are equal. Let z_j be a function which relies on the bits j, …, j+2t of x or y (the bits which the jth bit of z rely on). Now consider that we have a function which, if we invert x_j or y_j, ought to invert z_j. Thus $x_j = y_j$, and we can continue our argument down the line.

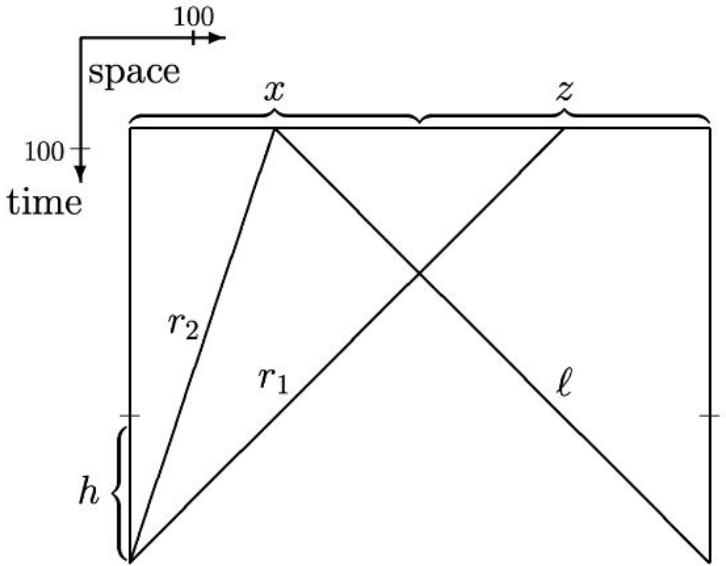
Cryptanalysis

Our Cryptanalysis will rely on **two** key observations:

1. Because our array is binary, propagations of changes do not necessarily occur at every time step for every bit positions.
2. A certain pattern has predictable behavior in our CA.

Cryptanalysis

Consider that for a CA as described above, at each timestep, a bit is only affected by the bits immediately to it's right and left. Often times we consider the dependence area of a bit, as shown below.



Cryptanalysis

Notice that for the repeating alternating pattern (010101…), the pattern propagates itself through the CA. It does this slowly, but it does so in a stable way. This will be the key to breaking our hash function.

Note that the pattern expands itself to the left at least three times as fast as it does to the right.

Collision Construction

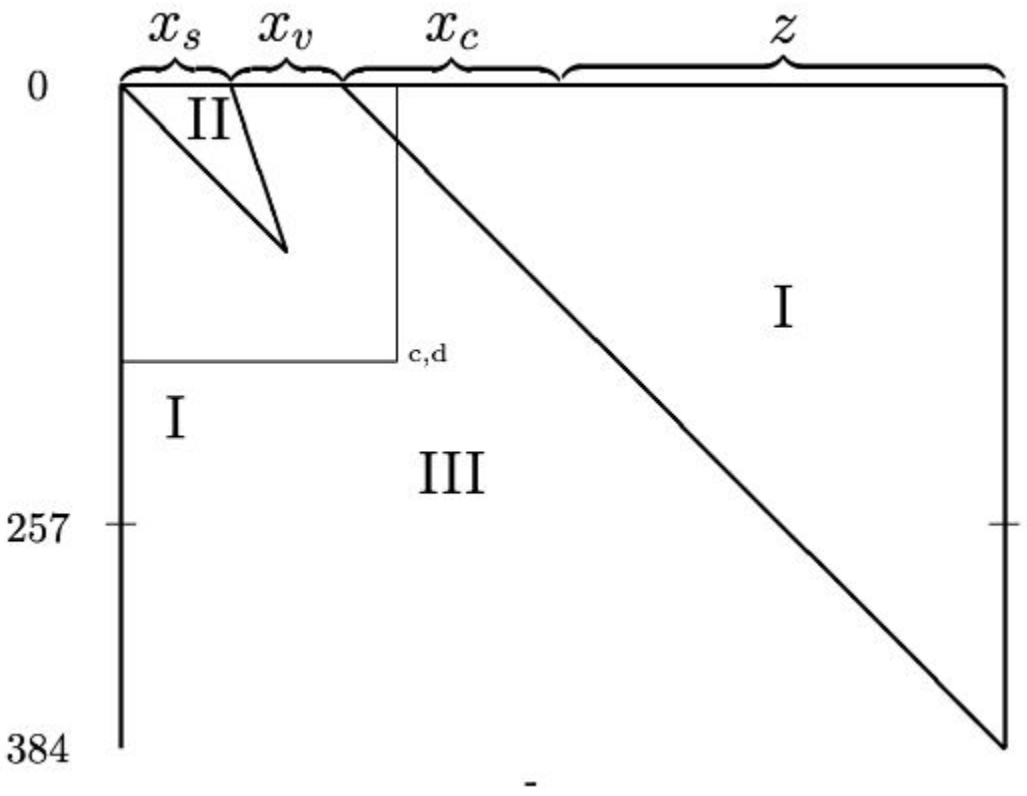
Our concrete scheme considers a 128-bit output (so d - c = 128).

We let c = 257 and d = 384. Our input should be 512 bits, z will be randomly chosen and does not matter, thus we must choose 256 bits for x. The authors let x be the concatenation of three substrings x_s, x_v, and x_c with the following properties:

- x_s will have the alternating pattern and it will be 64-bits.
- x_v need only be 62-bits.
- x_c need only be 130-bits.

With this construction we can show that for any two inputs such that x_s and x_c are equivalent, their output will collide. Thus they found a set of 2^{130} different bitstring which each have 2^{62} collisions.

Collision Construction



Conclusion

Examples like this, showed the danger of using only statistical methods of analysis on cryptographic schemes. Damgård's function was examined using statistical, non-cryptographic, analysis methods and showed very good results. Despite that, only 5 years later, people were able to come up with, not just one, but 2^{192} collisions! And an easy method for constructing any of these collisions.

Luca Mariot, points out in their paper "Insights gained after a decade of cellular automata-based cryptography", that often examples like this come from people in non-cryptographic fields putting fourth cryptographic schemes and analyzing them with the methods that they are familiar with. This provides concrete evidence as to the importance of the specific cryptographic standards which have been developed by the field.

References

[1] Mikkel Thorup. High speed hashing for integers and strings. CoRR, abs/1504.06804, 2015.

[2] Ivan Bjerre Damg˚ard. A design principle for hash functions. In Gilles Brassard, editor, Advances in Cryptology — CRYPTO' 89 Proceedings, pages 416–427, New York, NY, 1990. Springer New York.

[3] Joan Daemen, Ren´e Govaerts, and Joos Vandewalle. A framework for the design of one-way hash functions including cryptanalysis of damg˚ard's one-way function based on a cellular automaton. In Hideki Imai, Ronald L. Rivest, and Tsutomu Matsumoto, editors, Advances in Cryptology — ASIACRYPT '91, pages 82–96, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[4] Luca Mariot. Insights gained after a decade of cellular automata-based cryptography. In Maximilien Gadouleau and Alonso Castillo-Ramirez, editors, Cellular Automata and Discrete Complex Systems, pages 35–54, Cham, 2024. Springer Nature Switzerland.