Intro to Public Key Cryptography

CS/ECE 407

Today’s objectives

Articulate goals for public key cryptography

Discuss notion of structured hardness

Define the discrete logarithm/DDH assumptions

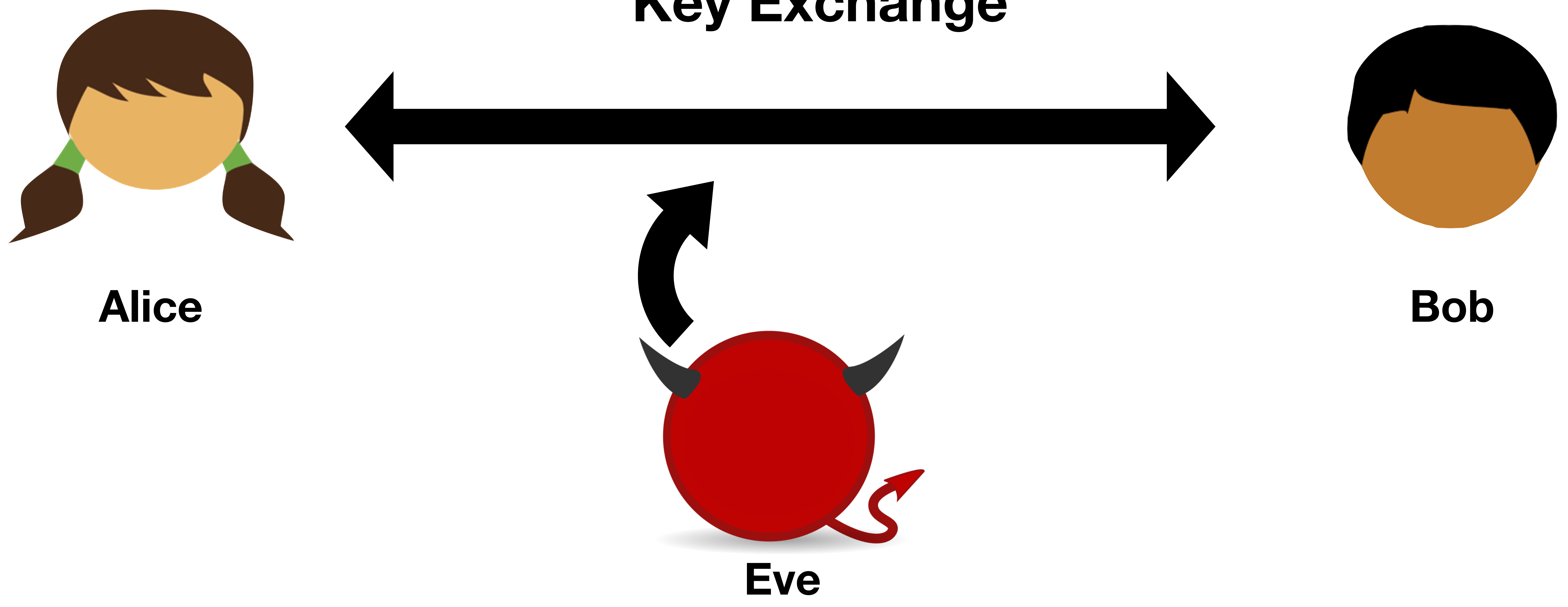
Demonstrate Diffie-Hellman Key Exchange

Alice
k
Eve
Bob
k

Public Key Cryptography:

Can Alice and Bob securely communicate, even if they are speaking for the very first time?

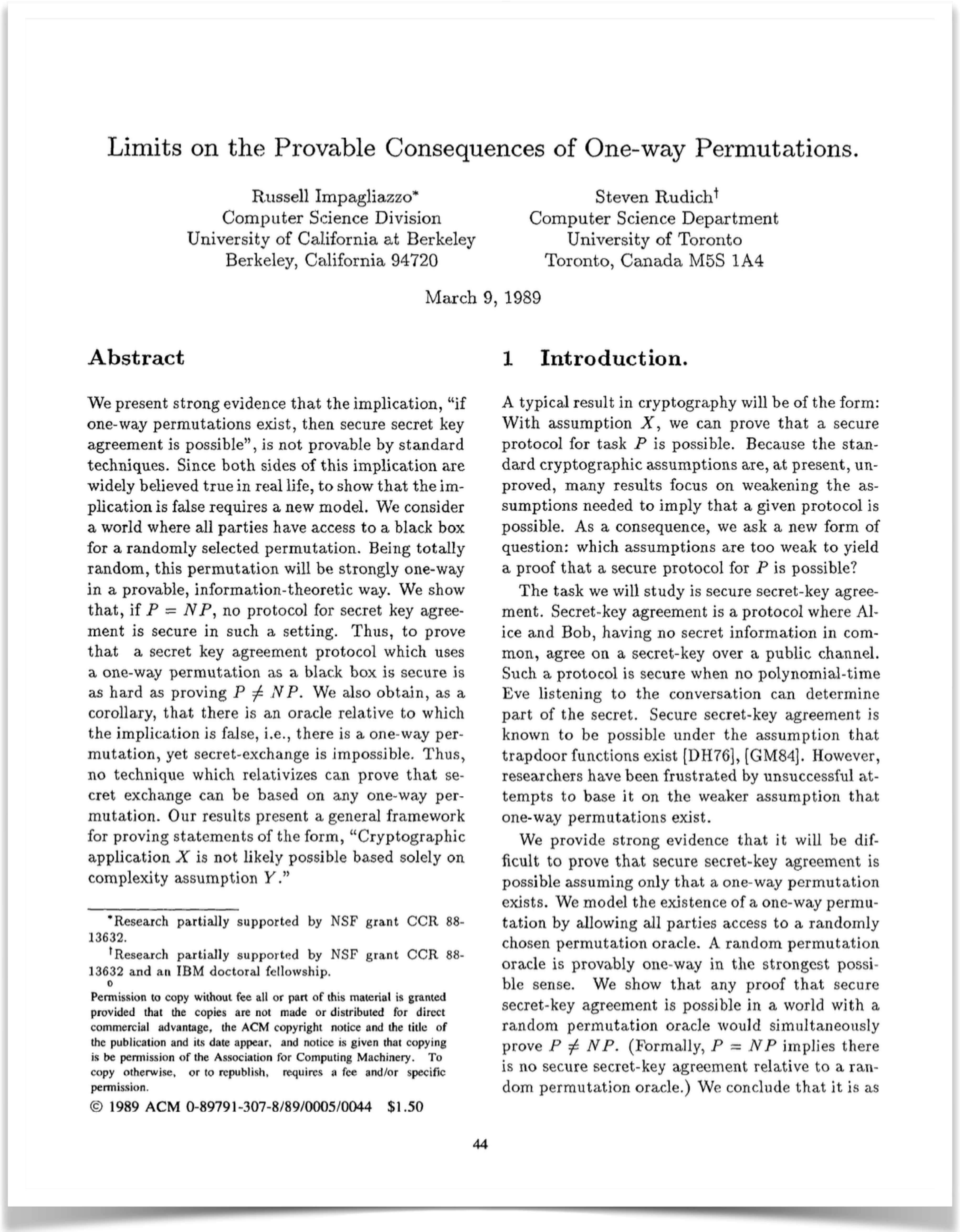
Key Exchange



Alice and Bob communicate a few times, then agree on a secret key k

Under a cryptographic assumption, Eve does not know k

Key Exchange



Limits on the Provable Consequences of One-way Permutations.

Russell Impagliazzo*
Computer Science Division
University of California at Berkeley
Berkeley, California 94720

Steven Rudich†
Computer Science Department
University of Toronto
Toronto, Canada M5S 1A4

March 9, 1989

Abstract

We present strong evidence that the implication, "if one-way permutations exist, then secure secret key agreement is possible", is not provable by standard techniques. Since both sides of this implication are widely believed true in real life, to show that the implication is false requires a new model. We consider a world where all parties have access to a black box for a randomly selected permutation. Being totally random, this permutation will be strongly one-way in a provable, information-theoretic way. We show that, if $P = NP$, no protocol for secret key agreement is secure in such a setting. Thus, to prove that a secret key agreement protocol which uses a one-way permutation as a black box is secure is as hard as proving $P \neq NP$. We also obtain, as a corollary, that there is an oracle relative to which the implication is false, i.e., there is a one-way permutation, yet secret-exchange is impossible. Thus, no technique which relativizes can prove that secret exchange can be based on any one-way permutation. Our results present a general framework for proving statements of the form, "Cryptographic application X is not likely possible based solely on complexity assumption Y."

*Research partially supported by NSF grant CCR 88-13632.

†Research partially supported by NSF grant CCR 88-13632 and an IBM doctoral fellowship.

1 Introduction.

A typical result in cryptography will be of the form: With assumption X, we can prove that a secure protocol for task P is possible. Because the standard cryptographic assumptions are, at present, unproved, many results focus on weakening the assumptions needed to imply that a given protocol is possible. As a consequence, we ask a new form of question: which assumptions are too weak to yield a proof that a secure protocol for P is possible?

The task we will study is secure secret-key agreement. Secret-key agreement is a protocol where Alice and Bob, having no secret information in common, agree on a secret-key over a public channel. Such a protocol is secure when no polynomial-time Eve listening to the conversation can determine part of the secret. Secure secret-key agreement is known to be possible under the assumption that trapdoor functions exist [DH76], [GM84]. However, researchers have been frustrated by unsuccessful attempts to base it on the weaker assumption that one-way permutations exist.

We provide strong evidence that it will be difficult to prove that secure secret-key agreement is possible assuming only that a one-way permutation exists. We model the existence of a one-way permutation by allowing all parties access to a randomly chosen permutation oracle. A random permutation oracle is provably one-way in the strongest possible sense. We show that any proof that secure secret-key agreement is possible in a world with a random permutation oracle would simultaneously prove $P \neq NP$. (Formally, $P = NP$ implies there is no secure secret-key agreement relative to a random permutation oracle.) We conclude that it is as

44

Key exchange does not exist given only a random oracle

And hence it doesn't exist *(probably)* under existence of only OWFs/PRGs/PRFs

Key Exchange

A poly-time computable function f is called a **one-way function** if for any PPT program A and for all inputs x the following probability is negligible (in λ):

$$\Pr\left[\ f(A(f(x))) = f(x)\ \middle|\ x \leftarrow \{0,1\}^{\lambda}\right]$$

"f is easy to compute, but hard to invert"

Key Exchange

A poly-time computable function f is called a **one-way function** if for any PPT program A and for all inputs x the following probability is negligible (in λ):

$$\Pr\left[\, f(A(f(x)) = f(x) \;\middle|\; x \leftarrow \{0,1\}^{\lambda} \right]$$

"f is easy to compute, but hard to invert"

Very informally, the reason OWFs *(probably)* don't give us key exchange is that a OWF alone has no *structure*

Key Exchange

A poly-time computable function f is called a **one-way function** if for any PPT program A and for all inputs x the following probability is negligible (in λ):

$$\Pr\left[\, f(A(f(x)) = f(x) \;\middle|\; x \leftarrow \{0,1\}^{\lambda} \right]$$

"f is easy to compute, but hard to invert"

Very informally, the reason OWFs *(probably)* don't give us key exchange is that a OWF alone has no *structure*

Let's find a specific OWF with some extra useful algebraic structure

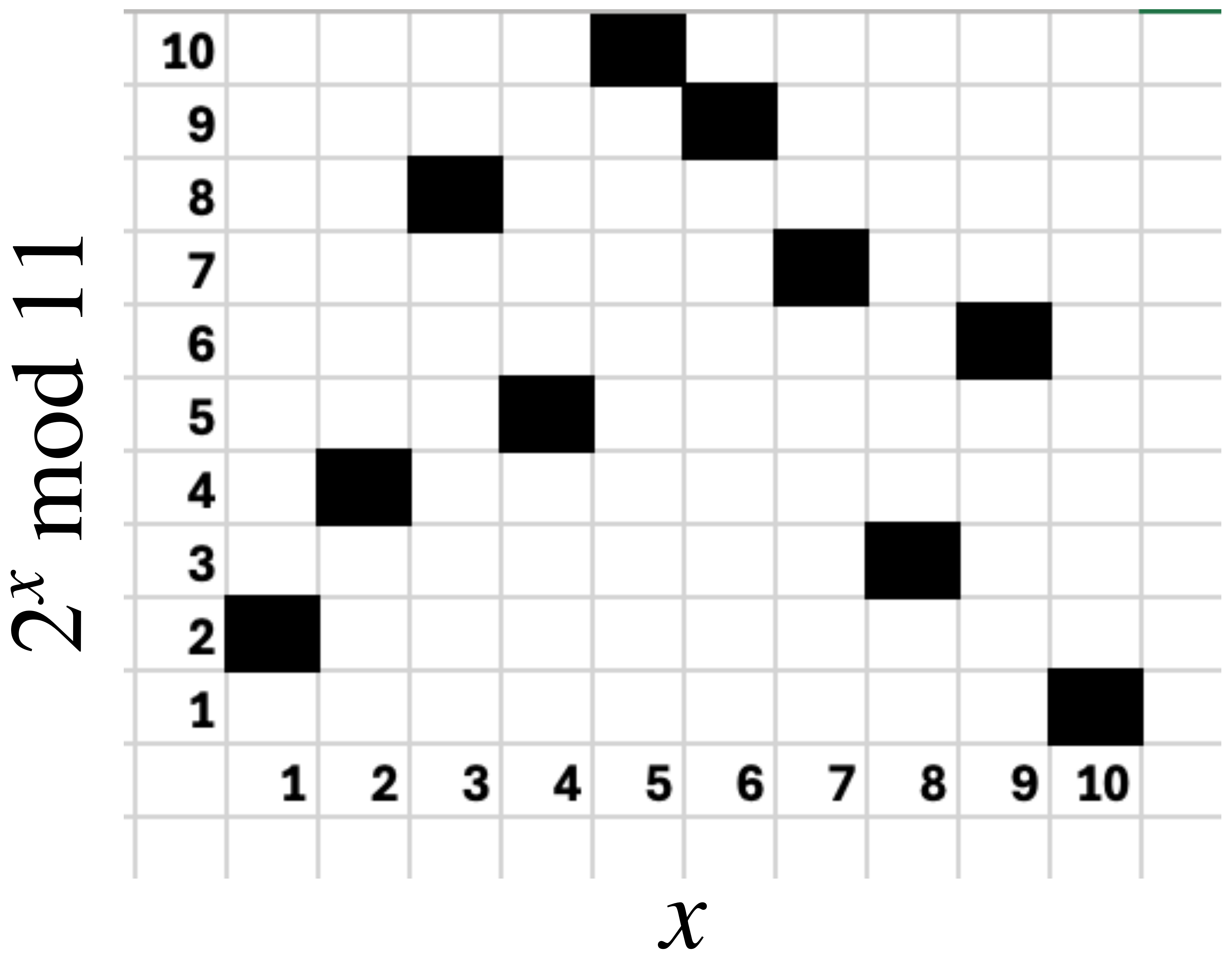
(We'll see what structure means shortly)

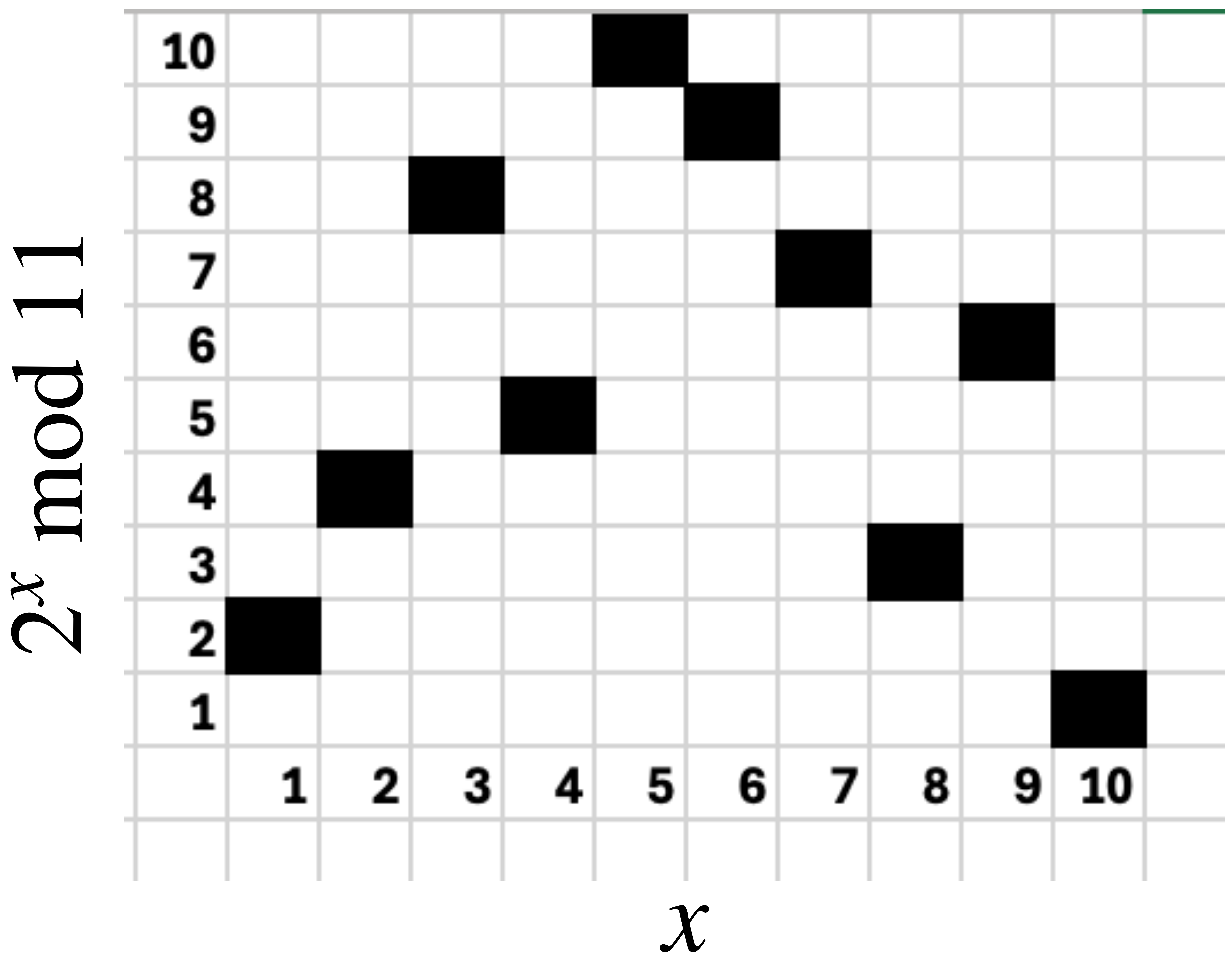
Modular Exponentiation

Let p be a prime, and consider the integers modulo p

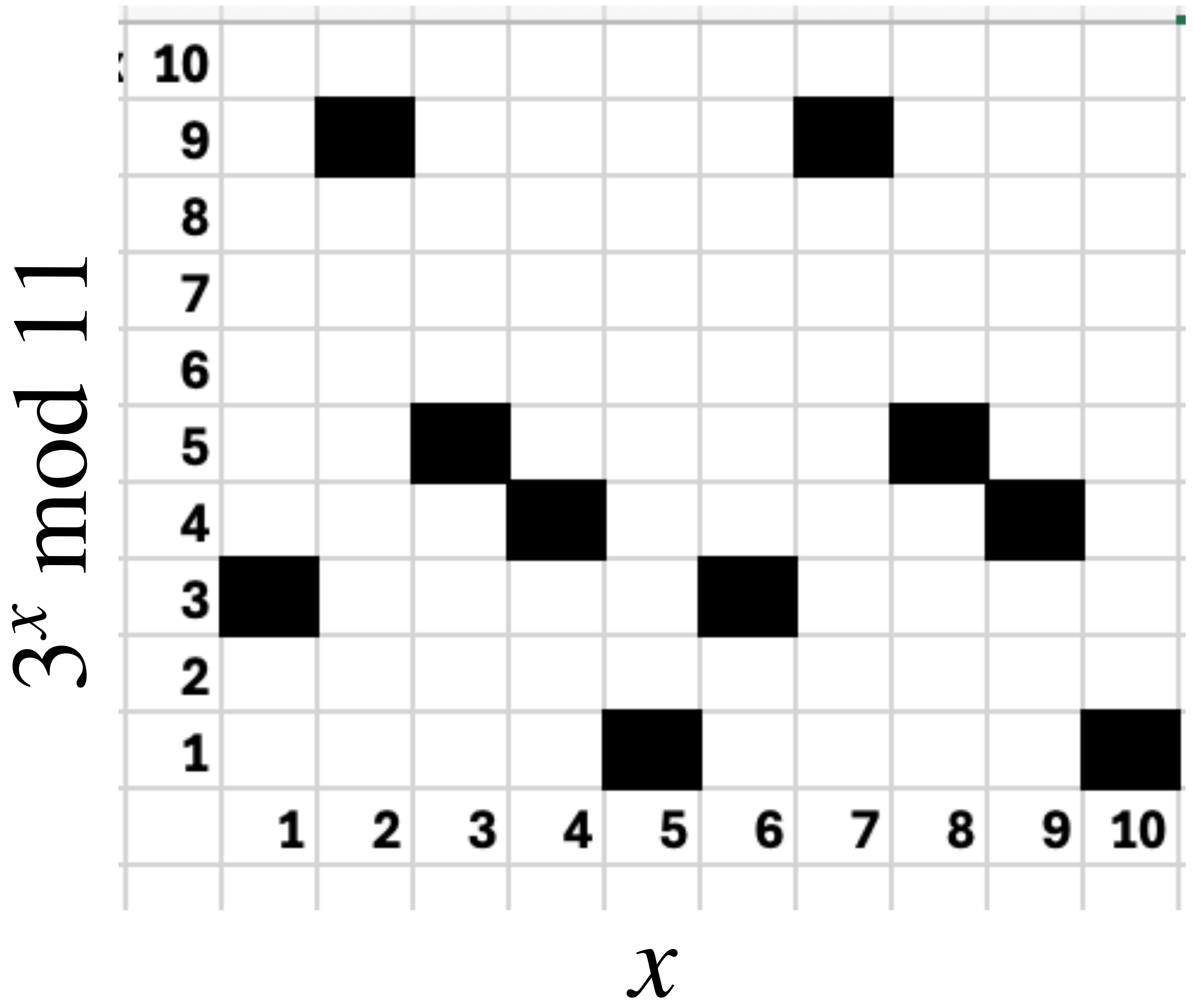
For instance, consider p = 11

> Consider raising some number, say 2 to various powers x, then reducing modulo 11

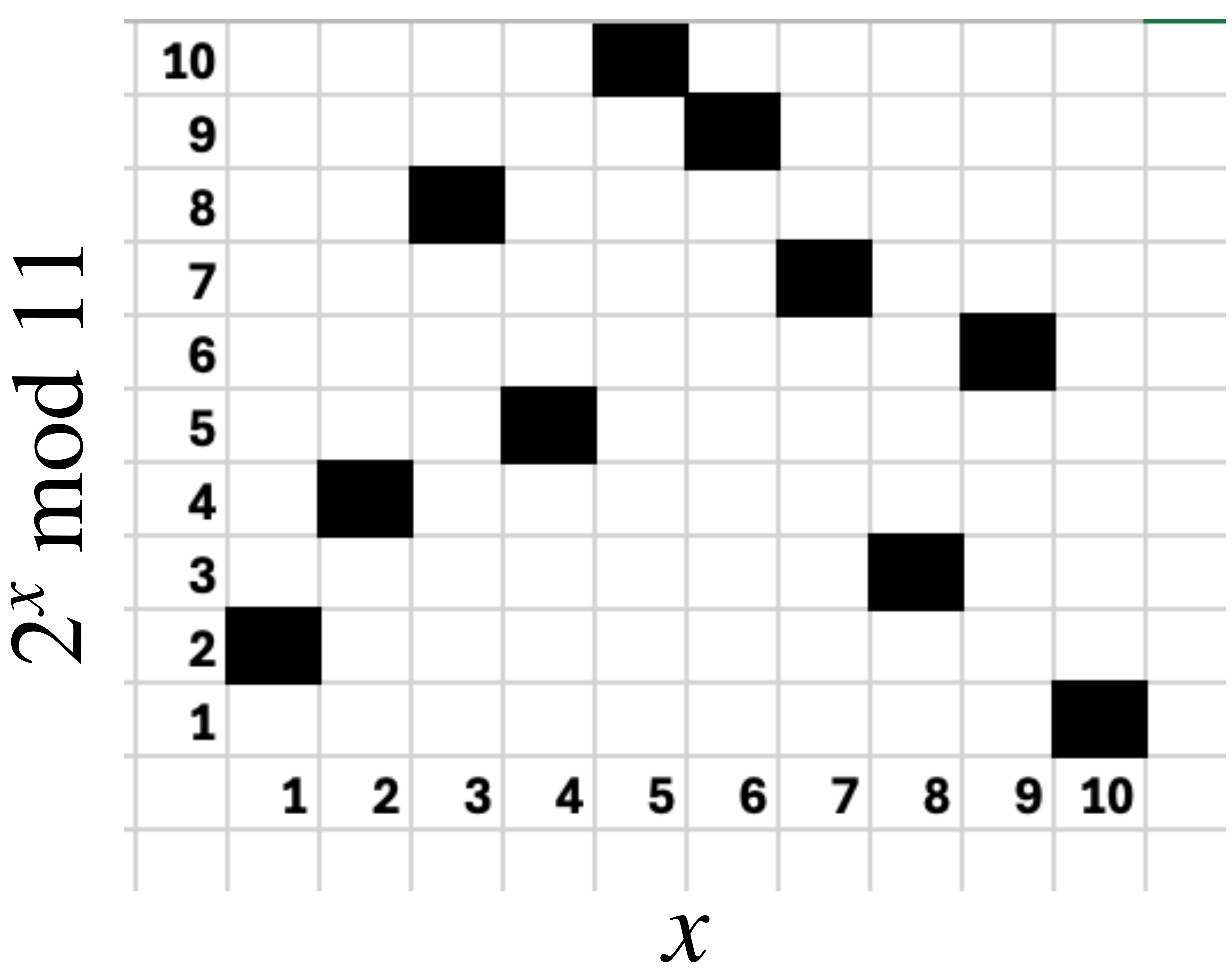

$2^x \bmod 11$
10
9
8
7
6
5
4
3
2
1
1
2
3
4
5
6
7
8
9
10
x

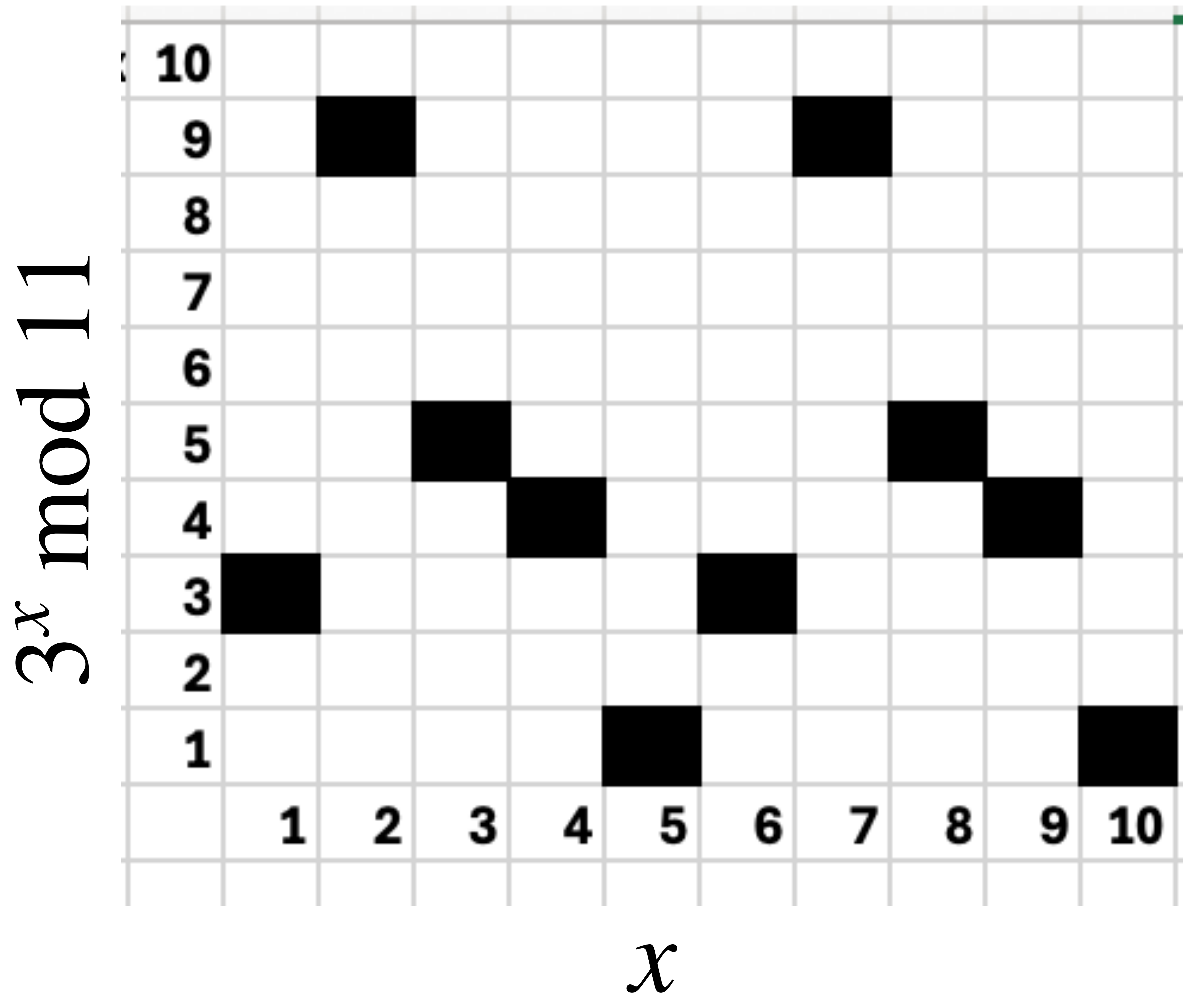


$2^x \bmod 11$
10
9
8
7
6
5
4
3
2
1
1
2
3
4
5
6
7
8
9
10
x



$3^x \bmod 11$
10
9
8
7
6
5
4
3
2
1
1
2
3
4
5
6
7
8
9
10
x





2^x hits every number 1,…,10

3^x does *not* hit every number 1,…,10

2 is called a **generator**

3 *is not* a generator

Modular Exponentiation

Given a prime with generator g and a number x, compute $g^x \bmod p$

Efficiently computable via repeated squaring

Modular Exponentiation

Given a prime with generator g and a number x, compute $g^x \bmod p$

Computable by: repeated squaring

Discrete Logarithm

Given a prime with generator g and $g^x \bmod p$, compute x

Modular Exponentiation

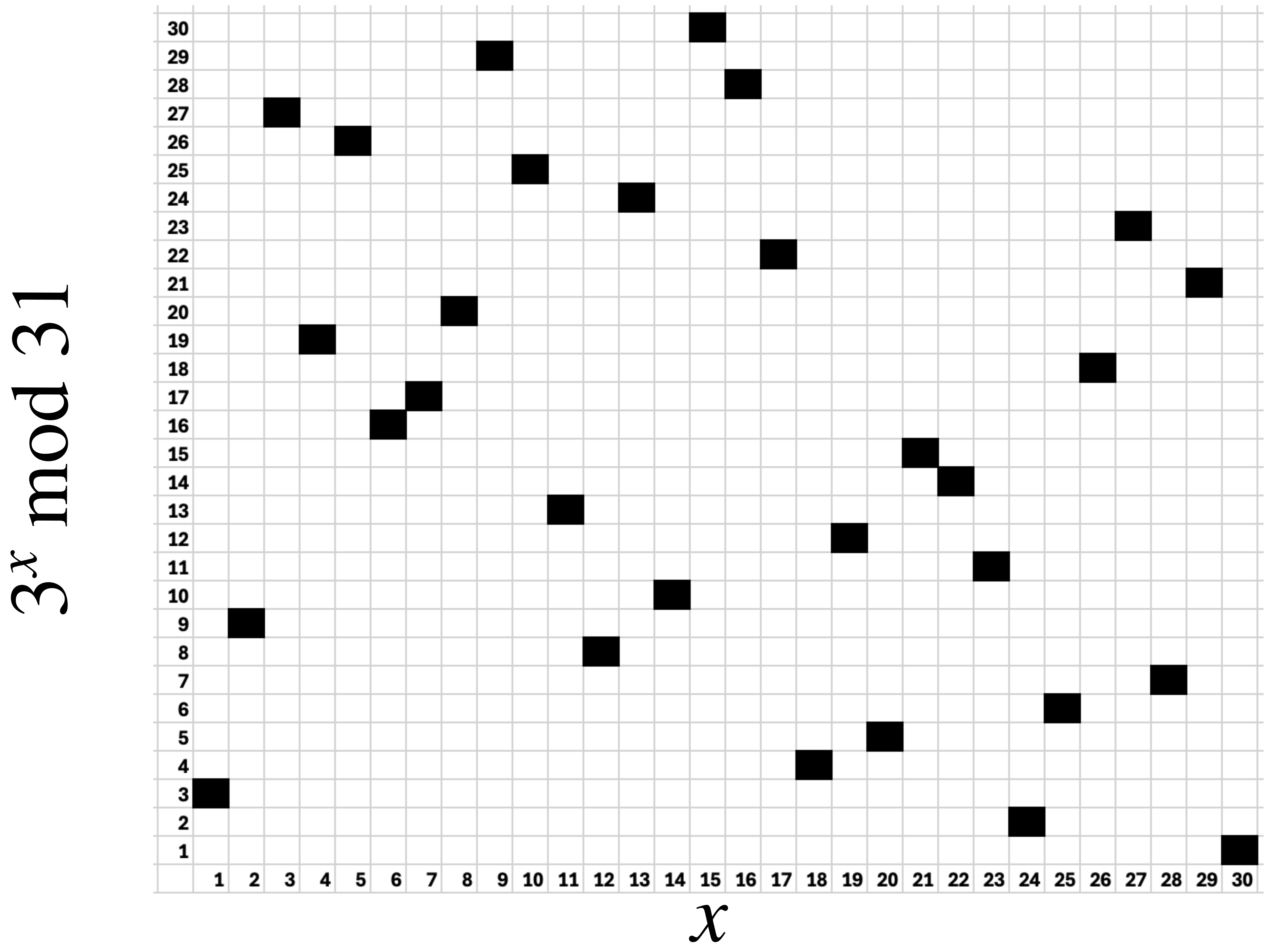
Given a prime with generator g and a number x, compute $g^x \bmod p$

Computable by: repeated squaring

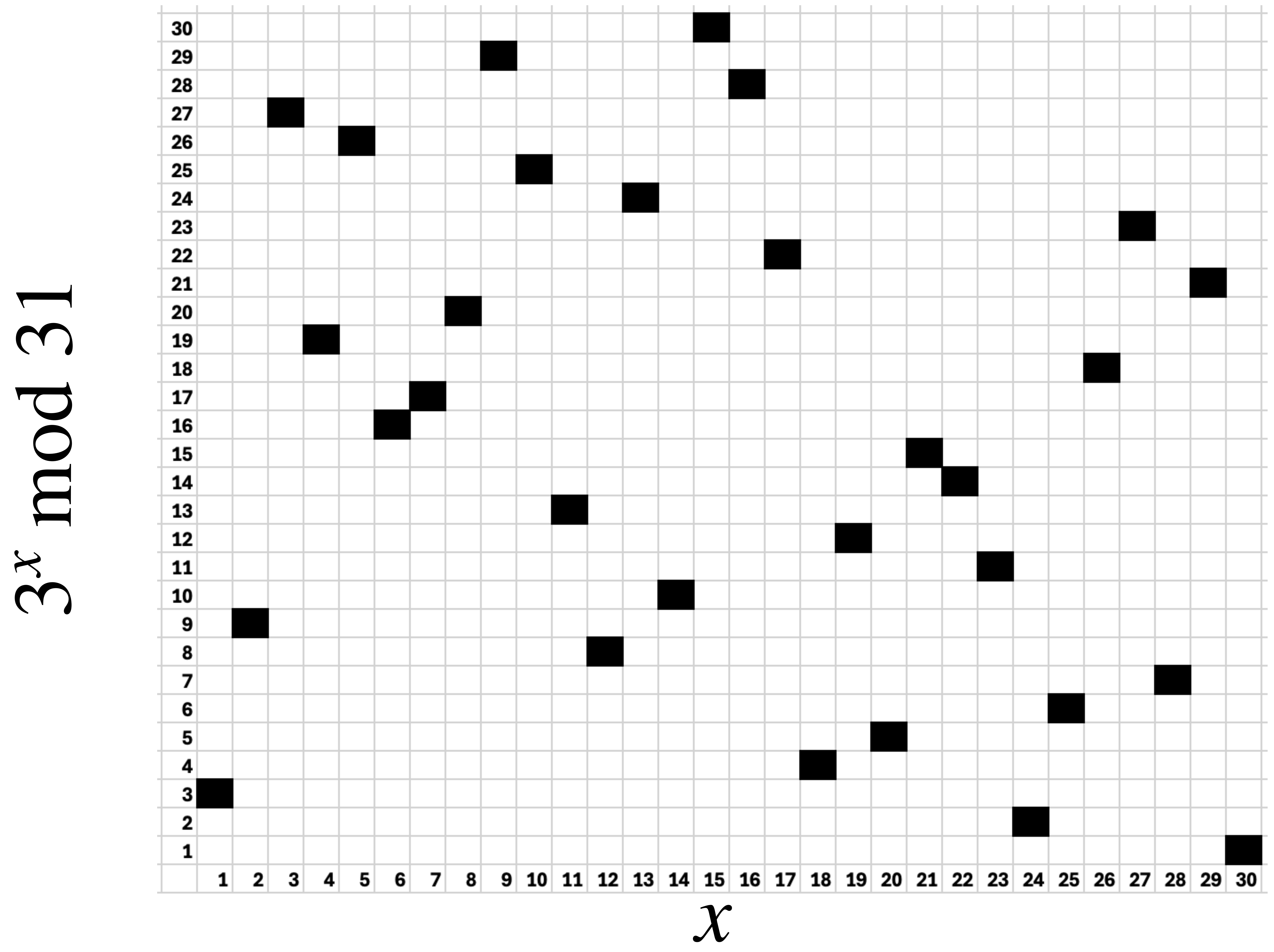
Discrete Logarithm

Given a prime with generator g and $g^x \bmod p$, compute x

Computable by: ???
we don’t have a poly time (classical) algorithm!



$3^x \bmod 31$
x



Intuition

If I tell you a number y, it seems to be challenging to find a number x such that $g^x \bmod p = y$

P =
24103124269210325885520760221975660748569505485
02459942654116941958108831682612228890093858261
34161467322714147790401219650364895705058263194
27307068050092230627347453410734066962460145893
61659774041027169249453200378729434170325843778
65919814376319377685986952408894019557734611984
35453015470437472077499697637500843089263392955
59968882457872412993810129130294592999947926365
26405928464720973038494721168143446471443848852
0940127459844288859336526896320919633919

A 2048-bit prime

With generator 2

Discrete Logarithm Assumption (for integers mod p)

Let Gen be an algorithm that defines a family of prime numbers and generators

We say the discrete logarithm is hard for that family if the following probability is negligible:

$$\Pr\left[A(p, g, g^x \bmod p) = x \;\middle|\; \begin{array}{c} (p, g) \leftarrow \mathrm{Gen}(\lambda) \\ x \leftarrow \{0, \ldots p-1\} \end{array} \right]$$

Modular Exponentiation

Interesting, because it seems to be one-way, **but it also has structure!**

$$\left(g^x\right)^y = g^{xy} = \left(g^y\right)^x$$

Decisional Diffie-Hellman Assumption (for integers mod p)

Let Gen be an algorithm that defines a family of prime numbers and generators

We say the DDH problem is hard for that family if the following ensembles are indistinguishable:

$$\left\{ (p, g, g^x, g^y, g^z) \;\middle|\; \begin{array}{l} (p, g) \leftarrow \mathrm{Gen}(\lambda) \\ x \leftarrow \{0, \ldots p-1\} \\ y \leftarrow \{0, \ldots p-1\} \\ z \leftarrow \{0, \ldots p-1\} \end{array} \right\} \approx \left\{ (p, g, g^x, g^y, g^{xy}) \;\middle|\; \begin{array}{l} (p, g) \leftarrow \mathrm{Gen}(\lambda) \\ x \leftarrow \{0, \ldots p-1\} \\ y \leftarrow \{0, \ldots p-1\} \end{array} \right\}$$

Definition: Group

A **group** is a set S with some operation ★ s.t.

$$x \bigstar (y \bigstar z) = (x \bigstar y) \bigstar z$$

And where there exists some element $\underline{\mathbf{1}}$

$$x \bigstar \underline{\mathbf{1}} = \underline{\mathbf{1}} \bigstar x = x$$

And where for every x there exists some x^{-1} s.t.

$$x \bigstar x^{-1} = \underline{\mathbf{1}}$$

Theorem [Euler]

$\mathbb{Z}_p^*$ is a *cyclic* group under multiplication:

Cyclic: There exists some **generator** g s.t.

$$\{ g^0, g^1, g^2, \dots g^{p-2} \} = \mathbb{Z}_p^*$$

Notation:

The order of a group is its number of elements

Decisional Diffie-Hellman Assumption (for integers mod p)

Let Gen be an algorithm that defines a family of prime numbers and generators

We say the DDH problem is hard for that family if the following ensembles are indistinguishable:

$$\left\{ (p, g, g^x, g^y, g^z) \;\middle|\; \begin{array}{l} (p,g) \leftarrow \mathrm{Gen}(\lambda) \\ x \leftarrow \{0,\ldots p-1\} \\ y \leftarrow \{0,\ldots p-1\} \\ z \leftarrow \{0,\ldots p-1\} \end{array} \right\} \approx \left\{ (p, g, g^x, g^y, g^{xy}) \;\middle|\; \begin{array}{l} (p,g) \leftarrow \mathrm{Gen}(\lambda) \\ x \leftarrow \{0,\ldots p-1\} \\ y \leftarrow \{0,\ldots p-1\} \end{array} \right\}$$

Decisional Diffie-Hellman Assumption

Let Gen be an algorithm that defines a family of cyclic groups and their generators

We say the DDH problem is hard for that family if the following ensembles are indistinguishable:

$$\left\{ (\mathbb{G}, g, g^x, g^y, g^z) \;\middle|\; \begin{array}{l} (\mathbb{G}, g) \leftarrow \mathrm{Gen}(\lambda) \\ x \leftarrow \{0, \ldots |\mathbb{G}| - 1\} \\ y \leftarrow \{0, \ldots |\mathbb{G}| - 1\} \\ z \leftarrow \{0, \ldots |\mathbb{G}| - 1\} \end{array} \right\} \approx \left\{ (\mathbb{G}, g, g^x, g^y, g^{xy}) \;\middle|\; \begin{array}{l} (\mathbb{G}, g) \leftarrow \mathrm{Gen}(\lambda) \\ x \leftarrow \{0, \ldots |\mathbb{G}| - 1\} \\ y \leftarrow \{0, \ldots |\mathbb{G}| - 1\} \end{array} \right\}$$

Where do we go from here?

DDH gives key exchange, but only for one specific assumption

- Are there other assumptions we can use to achieve security?
- Can we achieve shorter keys?
- What about malleability/authenticity?
- How do we interface with tools we've already built?
- What about quantum adversaries?

Today's objectives

Articulate goals for public key cryptography

Discuss notion of structured hardness

Define the discrete logarithm/DDH assumptions

Demonstrate Diffie-Hellman Key Exchange