Blockchains & Cryptocurrencies

Key Management & Threshold Cryptography

Instructor: Matthew Green & Abhishek Jain
Johns Hopkins University - Spring 2023

*Some slides based on NBFMG

Key Management

To spend a coin, you need to know:

* some info from the public blockchain, and
* the owner's secret signing key

<u>So it's all about key management.</u>

Goals (for Key management)

- **Availability**: You can spend your coins
- **Security**: Nobody else can spend your coins
- Convenience of use

Hot storage

online

Cold storage

offline

hot secret key(s)

cold address(es)

payments

cold secret key(s)

hot address(es)

Splitting and Sharing Keys

Secret sharing [Shamir]

(k,n)-secret sharing: Divide a secret value S into n shares $S_1,\ldots,S_n$ such that:

- **Correctness**: *Any* k shares can be used to reconstruct S
- **Privacy**: S is hidden given at most k-1 shares

Secret sharing [Shamir]

- **Share(S):** Output a tuple $S_1,\ldots,S_n$
- **Reconstruct($x_1,\ldots,x_k$):** Output a value S*

k-Privacy: For any (S,S'), and any subset X of < k indices, the following two distributions are statistically close:

$$\{(S_1,\ldots,S_n) \leftarrow Share(S) : (S_i | i \in X)\},$$

$$\{(S'_1,\ldots,S'_n) \leftarrow Share(S') : (S'_i | i \in X)\}.$$

Example: n=2, k=2

Example: n=2, k=2

Share(S):

$x_1 = (S+R) \bmod p$ $x_2 = (S+2R) \bmod p$

Example: n=2, k=2

Share(S):

$x_1 = (S+R) \bmod p$ $\quad$ $x_2 = (S+2R) \bmod p$

Reconstruct(x_1,x_2):

$(2x_1 - x_2) \bmod p = S$

Example: n=2, k=2

- p = a large prime
- S = secret in [0, P)
- R = random in [0, P)

Share(S):

x_1 = (S+R) mod p x_2 = (S+2R) mod p

Reconstruct(x_1,x_2):

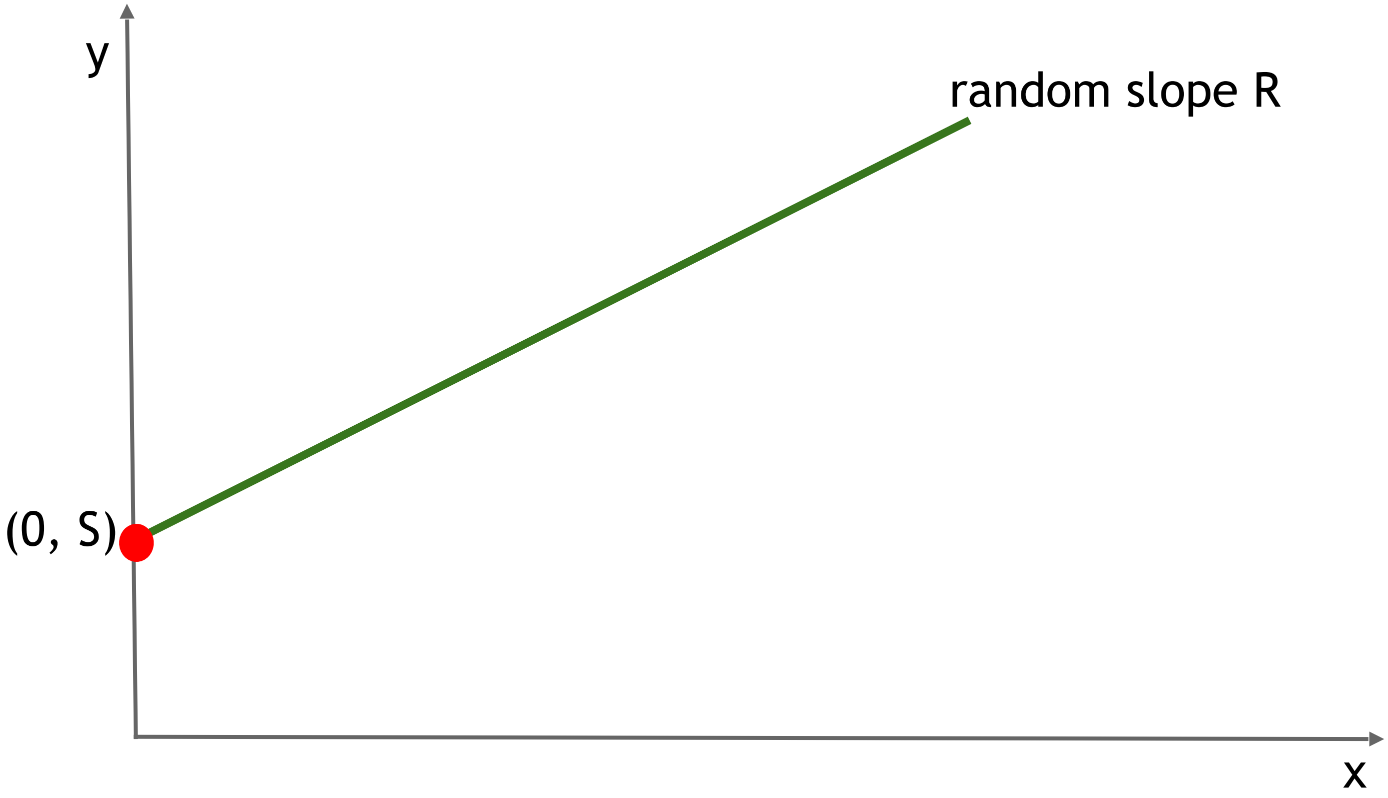
($2x_1$-x_2) mod p = S

<u>2-Privacy</u>: each x_i has uniform distribution over [0,P); independent of S

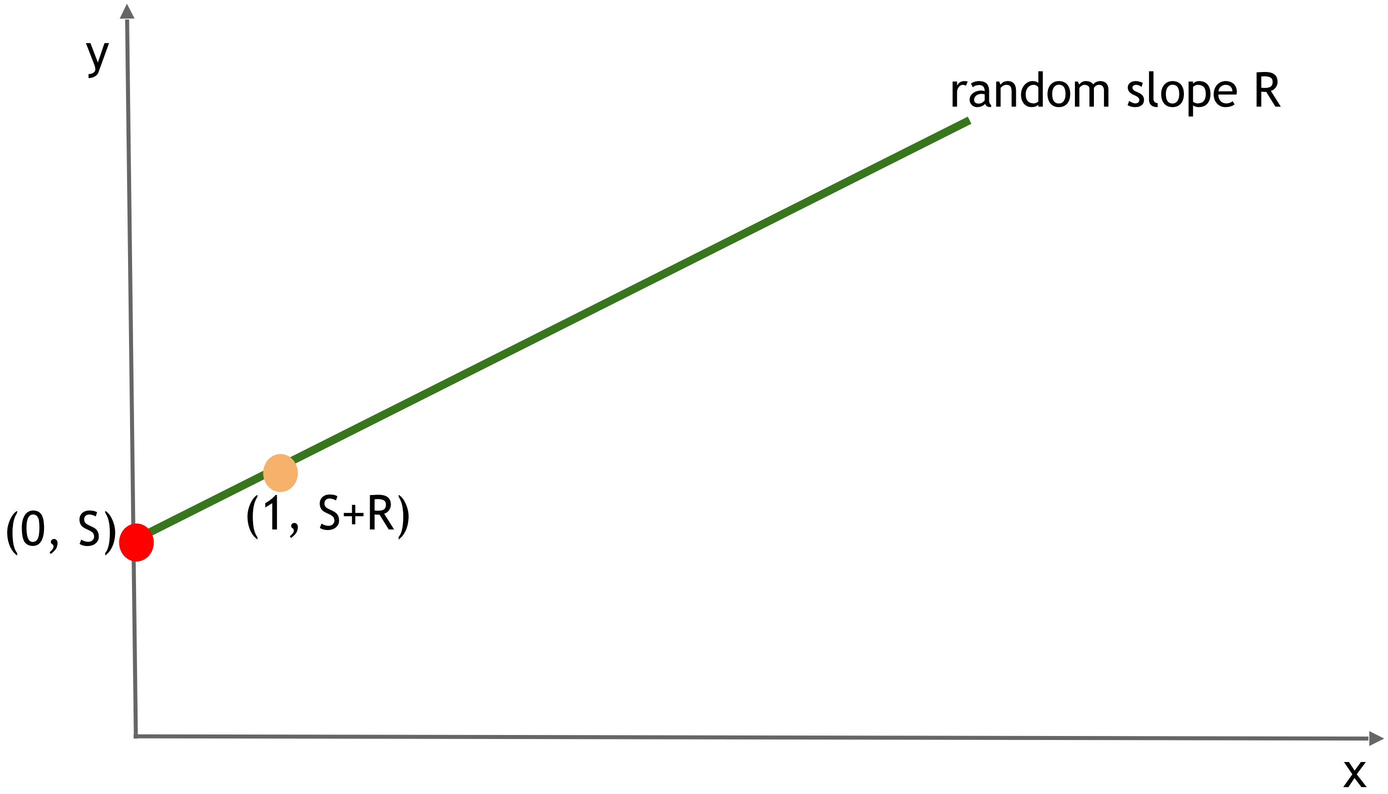
Example: k = 2, n > 2

Example: k = 2, n > 2

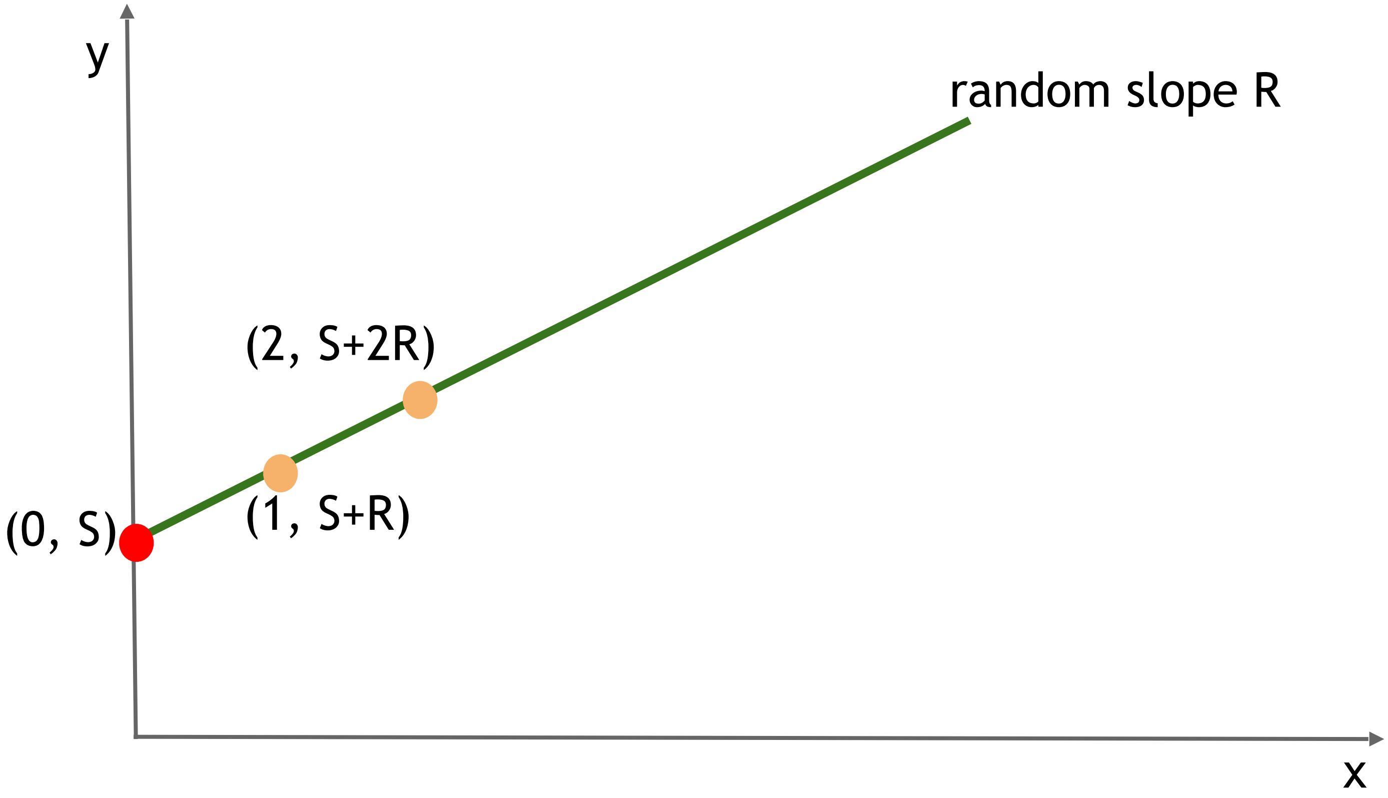
Example: k = 2, n > 2



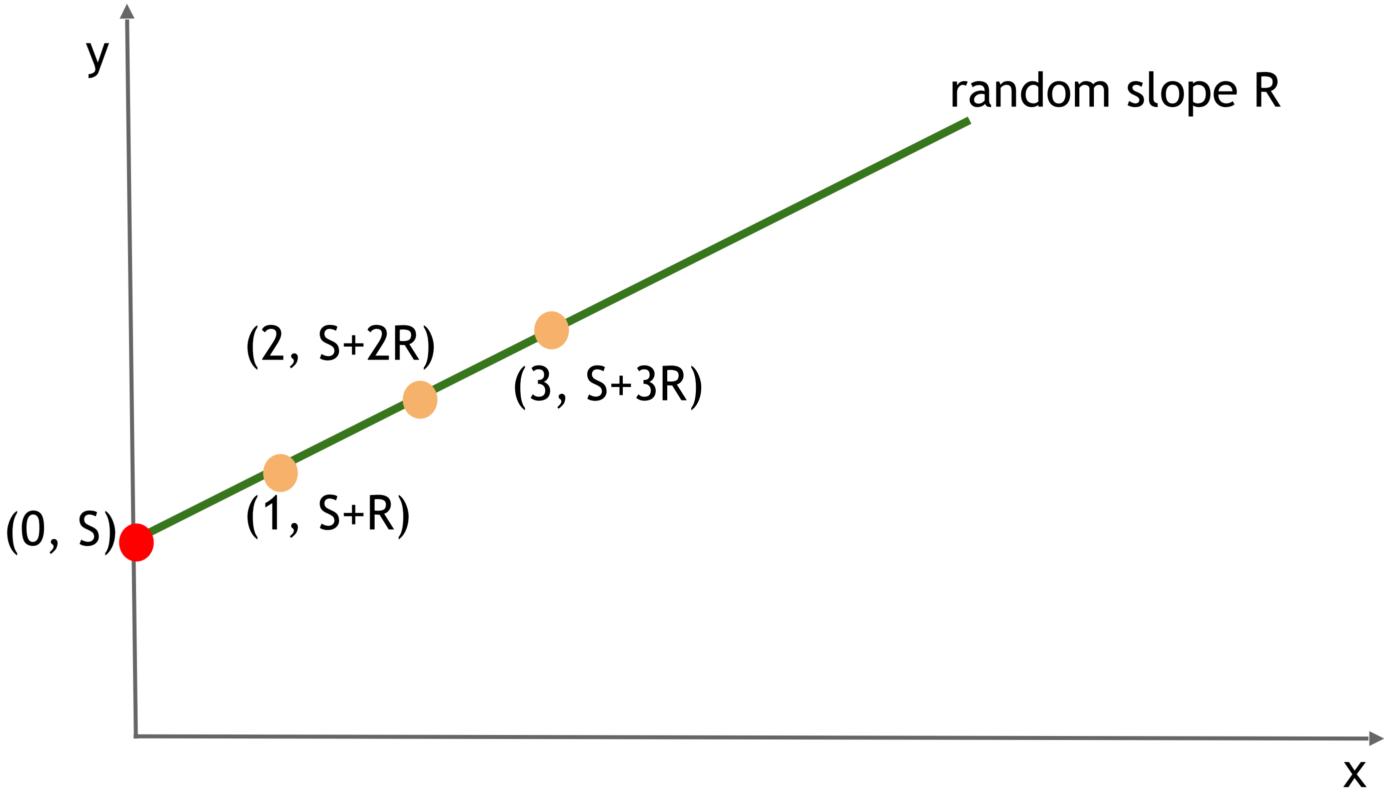
Example: k = 2, n > 2



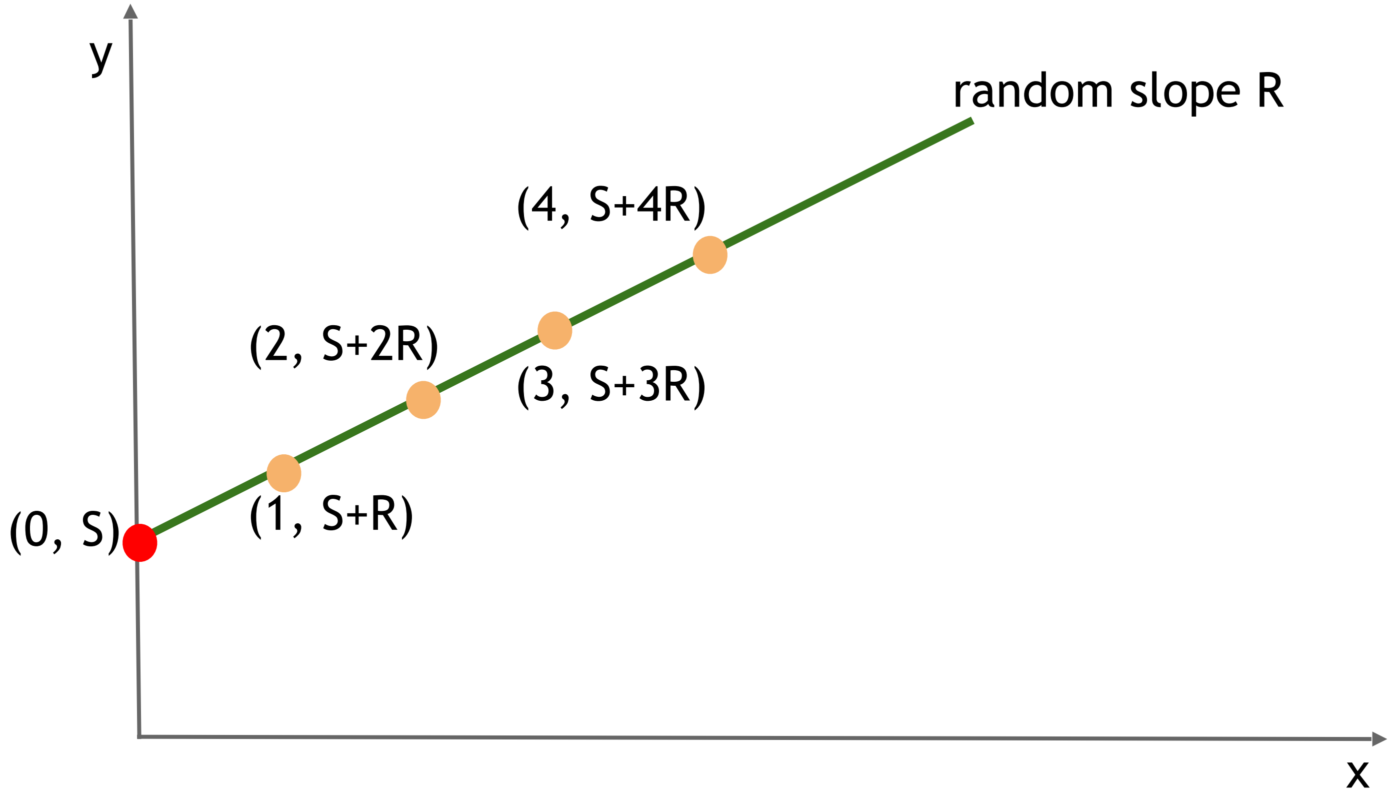
Example: k = 2, n > 2



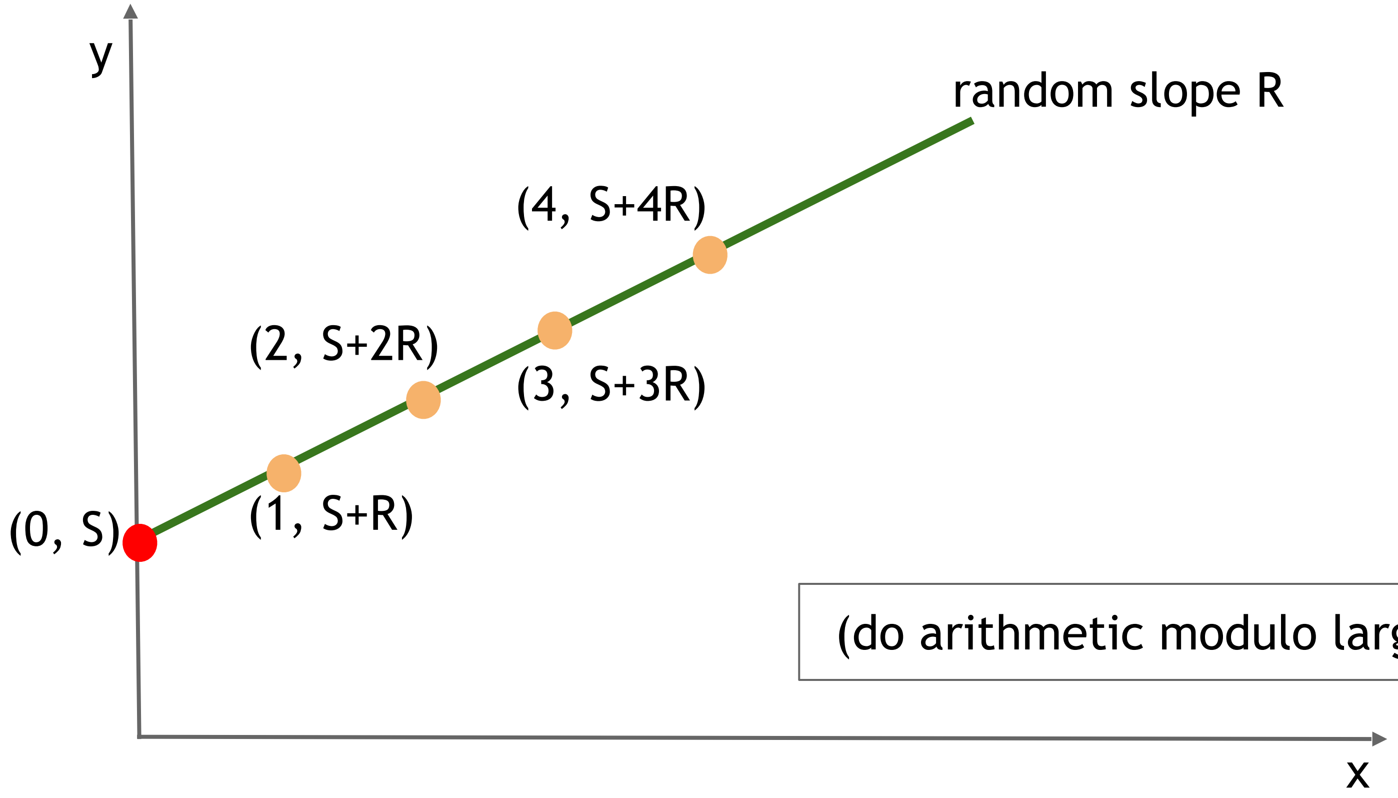
Example: k = 2, n > 2



Example: k = 2, n > 2

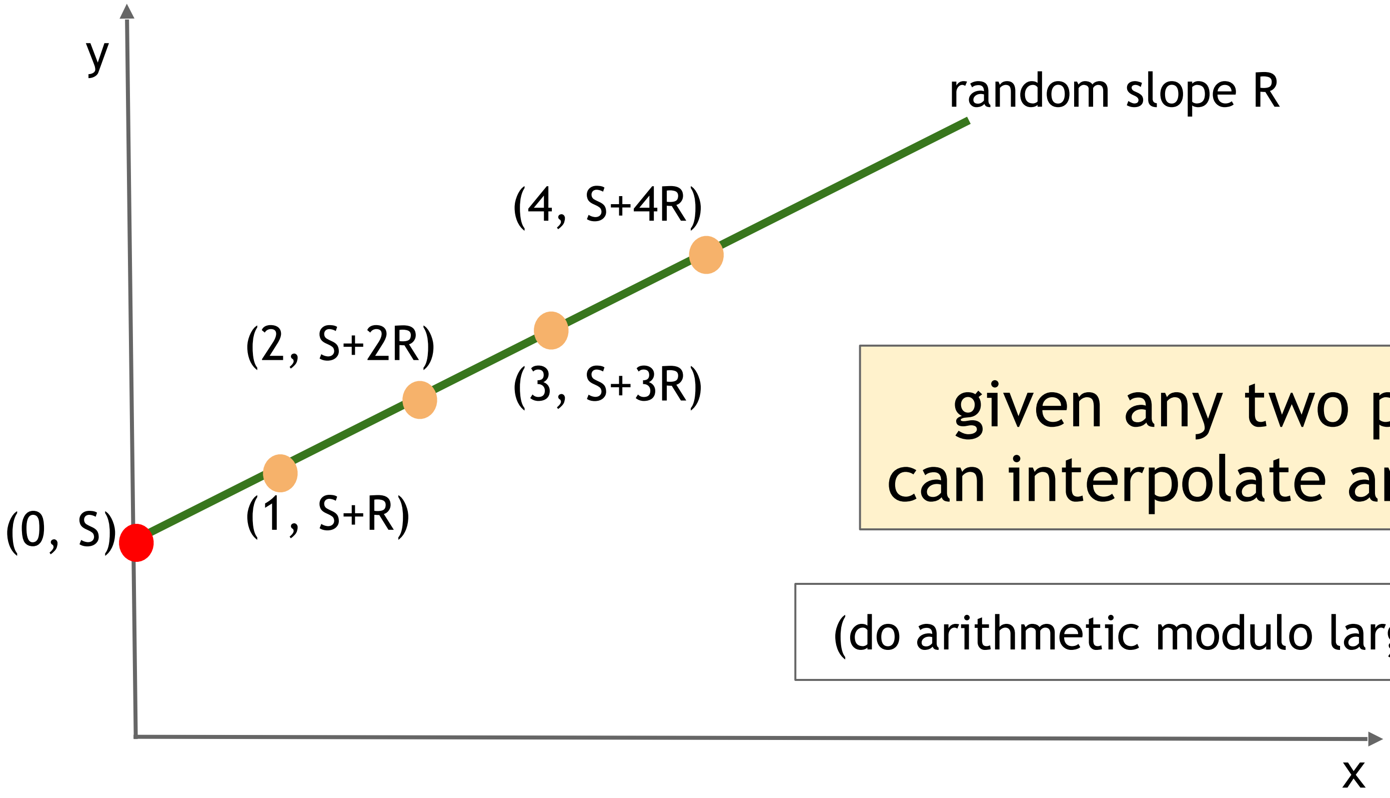


Example: k = 2, n > 2



(do arithmetic modulo large prime p)

Example: k = 2, n > 2

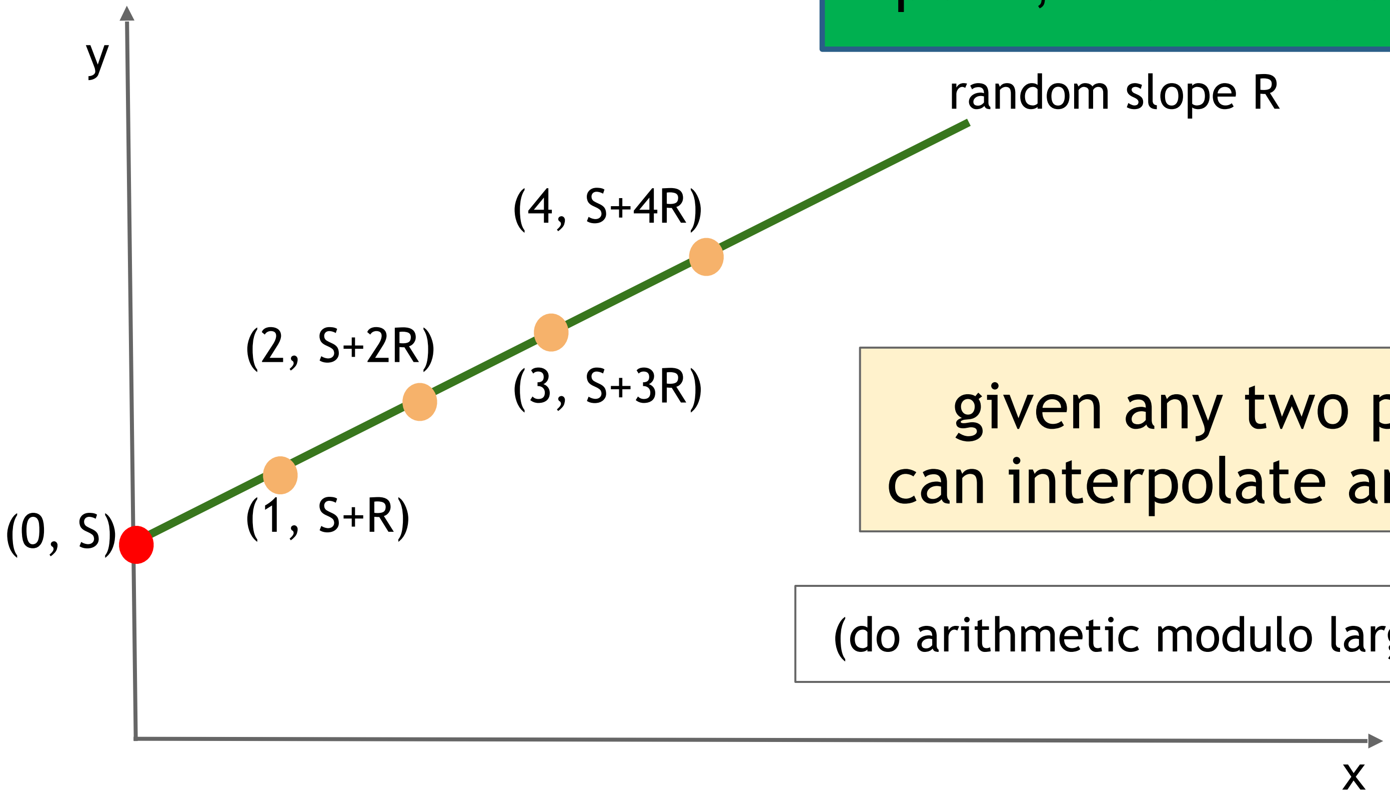


given any two points, can interpolate and find S

(do arithmetic modulo large prime p)

Example: k = 2, n > 2

k-Privacy: Given only one point, line is undetermined



given any two points, can interpolate and find S

(do arithmetic modulo large prime p)

Going Beyond $k = 2$

Going Beyond k = 2

Equation	Random parameters	Points needed to recover S
$(S + RX) \bmod p$	R	2
$(S + R_1X + R_2X^2) \bmod p$	R_1, R_2	3
$(S + R_1X + R_2X^2 + R_3X^3) \bmod p$	R_1, R_2, R_3	4

Going Beyond k = 2

Equation	Random parameters	Points needed to recover S
$(S + RX) \bmod p$	R	2
$(S + R_1X + R_2X^2) \bmod p$	R_1, R_2	3
$(S + R_1X + R_2X^2 + R_3X^3) \bmod p$	R_1, R_2, R_3	4
etc.		

Going Beyond k = 2

Equation	Random parameters	Points needed to recover S
$(S + RX) \bmod p$	R	2
$(S + R_1X + R_2X^2) \bmod p$	R_1, R_2	3
$(S + R_1X + R_2X^2 + R_3X^3) \bmod p$	R_1, R_2, R_3	4
etc.		

support K-out-of-N sharing, for any K, N

Secret sharing

Secret sharing

- **Good**: Store shares separately, adversary must compromise several shares to get the key.

- **Bad**: To sign, need to bring shares together, to first reconstruct the key. Point of vulnerability

Threshold Signatures

- **<u>(k,n)-Threshold Signatures</u>:** A signing key can be "divided" amongst n signers such that any subset of k signers can jointly produce a signature, but any subset of <k signers cannot
    - TSetup(1^n): Each party learns PK. Party i additionally learns Sk_i
    - TSign(m): Parties run a protocol to compute a signature **sig** on m
    - TVerify(PK,m,**sig**): Output 0/1

Threshold Signatures

- Advantages over Multi-Sig (that we saw earlier in class):
  - Threshold policy enforced in signature scheme as opposed to script
  - Threshold signature size the same as a single signature (as opposed to increasing linearly with k)
  - Threshold policy can be hidden in Threshold signatures
  - …

Threshold Signature Variants

- Confusingly, there is also a primitive called Multisignatures that is different from Multi-Sig
- Multisignatures are essentially n-out-of-n Threshold Signatures
- (t,n)-Threshold Signatures require a distributed key generation protocol. Multisignatures do not.

How to build Threshold Signatures

- Actively studied area for last 2-3 decades
- Many constructions, from many different assumptions, with various performance trade-offs
- In general, any signature scheme can be converted into threshold signatures using secure multiparty computation (MPC). But generic constructions can be expensive.

Today

- Some popular signature schemes: BLS (Boneh-Lynn-Shacham), Schnorr.
- How to “thresholdize” these signature schemes
- Differences involved due to deterministic vs randomized signing procedures
- Note: This is still an active area of research! Efforts also underway to standardize threshold signature schemes!